Energy Based Dissolution Simulation using Smoothed Particle Hydrodynamic Sampling

Min Jiang

A thesis submitted in partial fulfilment of the requirements of Bournemouth University for the degree of Doctor of Philosophy

April, 2016

Contents

List of Figures

List of Tables

List of Symbols

$\mathbf{v}, \mathbf{v}'$ velocity, updated velocity

$\mathbf{v}_l$ linear

$\mathbf{x}, \mathbf{x}'$ position, updated position

$\mathbf{a}$ acceleration

$\mathbf{g}$ gravity

$\mathbf{f}$ force density

$\mathbf{F}, \overline{\mathbf{F}}$ force, average force

$\mathbf{n}$ normal

$\mathbf{r}$ distance vector between points

$W(\mathbf{r})$ kernel function

$s(\mathbf{x})$ size function

$c(\mathbf{x})$ color field

ρ, ρ_0 density, rest density

p pressure

h smooth length

m, m_{vs} mass, volume mass

V volume

S surface area

N number of points

d average adjacent distance

θ temperature

t time step

λ density difference

k gas constant

μ viscosity coefficient

η dynamic viscosity coefficient

$\sigma, \boldsymbol{\sigma}$ tension coefficient, stress tensor

l threshold parameter

Φ domain

$\nu_{\mathbf{eff}}$ effective Nyquist frequency

Ω oscillation

T dissolution time

E_s excitation energy

E_0 activation energy

A attribute value

e Euler's number

n subscript of time-step

i, j, k, b, s, f subscript of particle index

$\mathbf{f}^B, \mathbf{f}^S$ force density of body force, surface force

$\mathbf{f}^p, \mathbf{f}^v, \mathbf{f}^t, \mathbf{f}^g$ force density of pressure, viscosity, surface tension, gravity

$\mathbf{F}^{\text{pres}}, \mathbf{F}^{\text{cohe}}, \mathbf{F}^{\text{surf}}$ pressure, cohesion, correction force

Abstract

Fluid simulation plays an important role in Computer Graphics and has wide applications in film and games. The desire for an improved physically-based fluid simulation solver has grown hand in hand with the advances made in Computer Graphics. Interesting fluid behaviours emerge when solid objects are added to a simulation: when fluid and solid make contact, they do not only have a physical interaction (e.g., buoyancy), but also a chemical reaction (e.g., dissolution) under the right conditions. Dissolution is one of the most common natural phenomena which is an important visual effect in fluid simulation. However this phenomenon is difficult to simulate due to the complexity of the behaviour and there are only few techniques available.

A novel unified particle-based method for approximating chemical dissolution is introduced in this thesis which is fast, predictable and visually plausible. The dissolution algorithm is derived using chemical Collision Theory and integrated into a Smoothed Particle Hydrodynamics (SPH) framework. The Collision Theory of chemistry is used as an analogy to the dissolution process modelling. Dissolution occurs when solute submerges into solvent. Physical laws govern the local excitation of solute particles based on the relative motion with solvent particles. When the local excitation energy exceeds a user specified threshold (activation energy), the particle will be dislodged from the solid. Unlike previous methods, this dissolution model ensures that the dissolution result is independent of solute sampling resolution. A mathematical relationship is also established between the activation energy, the interfacial surface area, and the total dissolution time — allowing for intuitive artistic control over the global dissolution rate. Applications of this method are demonstrated using a number of practical examples, including antacid pills dissolving in water and hydraulic erosion of non-homogeneous terrains.

Both solutes and solvents are represented by particles, and the distribution of the solute particles greatly affects the plausibility of the dissolution simulation. An even but stochastic distribution of particles

on both the surface and within the volume of the solute is essential for a good visual simulation of the dynamic process of dissolution.

A new iterative particle-based sampling method derived from SPH is introduced in this thesis which can generate a range of blue noise patterns and is computationally efficient, controllable and has a variety of applications. This approach resolves many of the limitations of classic blue noise methods, such as the lack of controllability or varying the distribution properties of the generated samples. Fast sampling is achieved in general dimensions for curves, surfaces and volumes. By varying a single parameter, the proposed method can generate a range of controllable blue noise samples with different distribution properties which are suitable for various applications such as adaptive sampling and multi-class sampling.

The SPH sampling approach is used for solute particle distribution which guarantees a predictable and smooth dissolution process thanks to the evenly distributed density and also gives the user control of the volume change during the phase transition. The proposed SPH sampling method achieves better visual effects compared with simple grid sampling and other blue noise sampling methods.

Our energy based dissolution simulation with SPH sampled solute and solvent ensures that the dissolution behaviour is physically and chemically plausible, while supporting features such as object separation and sharp feature rounding. The simulation is parallelized per particle on a GPU to enhance the performance.

Acknowledgements

First of all, I would like to express my sincere gratitude to Dr Richard Southern and Prof. Jian Jun Zhang for the excellent supervision. I have been extremely lucky to have had them as mentors who have provided continued supports and advices about my work.

I wish to express my gratitude to Prof. Rui Wang from University of Massachusetts and Dr Nadir Akinci from Massachusetts Institute of Technology, for their insight in the world of sampling and fluid simulation.

I would like to thank Jan Lewis who handled all the administrative tasks, cared my entire student life in Bournemouth University. I would also like to acknowledge with gratitude the support from graduate school.

Finally, I would like to thank my family and friends for backing me up during all the ups and downs of my research. I would also like to express my gratitude to the colleagues at the NCCA, for the amazing time I had during my PhD at Bournemouth University.

Gratitude also goes to the Chinese Scholarship Council for PhD studentships, which made it possible for me to undertake this research.

Declaration

This report has been created by myself and has not been submitted in any previous application for any degree. The work in this report has been undertaken by myself except where otherwise stated.

The materials related to dissolution simulation have been published in Jiang *et al.* (2015a) and Jiang *et al.* (2013). The work related to blue noise sampling and its applications have been published in Jiang *et al.* (2015b). The full list of publication is shown in Appendix A.

Chapter 1

Introduction

1.1 Background

Figure 1.1: *Fluid visual effect in film* Interstellar *(reproduced from Rigal (2015)).*

The simulation of natural phenomena is one of the most studied areas in Computer Graphics and enables animators to generate complex and physically plausible dynamics which would be impossible to create manually. Animating fluids like water, smoke, and fire using physically-based simulation is increasingly important in visual effects and makes great impacts in film and games. A visual effect of wave simulation in film 'Interstellar' is shown in the image above. Interactions among fluids and solids are frequently observed around our daily life and simulated for various purposes. However, real-time simulation of dissolution phenomena, including erosion and liquefying, is still to be developed due to the complexity of the behaviour model. This thesis goes into the interaction between fluid and solid, dealing with outstanding

issues in the dissolution simulation.

To achieve the complex dissolution behaviours, a unified particle representation is introduced for both fluid solvent and solid solute. The dissolution model is derived from chemical **Collision Theory** which explains how chemical reactions take place and why chemical reaction rates differ (Trautz 1916). Each solid particle carries the property of the local **Excitation Energy** which is the summed kinetic energy due to the neighbouring interactions and random thermal motions. The particle will detach from the solid when the energy exceeds a user specified threshold (activation energy). Unlike previous methods, our energy calculation ensures that the dissolution result is independent of solute sampling resolution by using volume–corrected mass instead of mass. A mathematical relationship is also established between the total dissolution time and the activation energy which provides the artists a control of the global dissolution rate while maintaining the physical plausibility of the simulation.

To promise a predictable dissolution process and avoid the problem of the fluid penetrating the boundary of the solute, an even distribution of the solute particles is preferable. To avoid discontinuities and artefacts during the particle phase transition, a novel blue noise sampling method — SPH sampling is proposed for solute particle distribution. It is experimentally demonstrated that the proposed SPH sampling method achieves better visual effects than random sampling or simple grid sampling and also gives more control over volume change during phase transition than other blue noise sampling methods.

This sampling method turns out to be very efficient and flexible. It conquers several limitations of the classic sampling methods, such as the lack of controllability or extension to general dimensions. By tuning one parameter, our SPH sampling can generate controllable blue noise patterns with different distribution properties and has a variety of applications outside the dissolution field such as image stippling and multi-class sampling.

1.1.1 Dissolution Simulation

Currently most work in fluid simulation focused on the behaviour of fluid itself, such as spray, waves and bubbles (Ihmsen *et al.* 2012; Cleary *et al.* 2007; Busaryev *et al.* 2012a) or the interaction with solids (Batty *et al.* 2007; Robinson-Mosher *et al.* 2008; Schechter and Bridson 2012; Akinci *et al.* 2013). Comparatively few researchers have tackled the challenging problem of the chemical reaction between fluid and solids, which, under the right conditions, can lead to solid mixing with the liquid: a phenomenon known as *dissolution*. The solid in this case is called the *solute* and the liquid is the *solvent*. Dissolution is one of the fundamental natural processes on earth, such as an antacid tablet dissolving into water. Bubbles are generated by a chemical reaction and the tablet is shrunk by mass transfer, shown in Fig. 1.2. Dissolution tests are widely used for food security (Kravtchenko *et al.* 1999) and in the pharmaceutical industry (CDER 2004). However due to the complex physical and chemical behaviours of dissolution simulation there are few publications currently exploring this problem.

Figure 1.2: *Tablet dissolving in water (Reproduced from Seltzerl (2013)).*

In this thesis, an energy–based dissolution model is proposed to describe this complicated fluid–solid interaction. Dissolution is a result of physical and chemical processes: our dissolution model follows *Collision Theory* which explains how the chemical reaction happens. It is used as an analogy to the dissolution process modelling. An energy function is introduced to measure the particle excitation of the solute. The concept of *volume–corrected mass* is proposed for the energy calculation, which ensures the consistent dissolution results even when the object sampling resolutions are different. This property of sampling resolution

independence is advantageous when performing pre-visualization of the dissolution process, allowing animators to quickly preview the dissolution behaviour with a low resolution sampling of the solute.

A solute particle will become dislodged from the solid if the local energy exceeds user specified *activation energy*. While physically justified, this threshold does not provide a user with an intuition of the total dissolution time. It is found that the activation energy can be approximated as a function of the interfacial surface area of the solvent/solute and the total dissolution time, providing animators with a tool to more accurately predict the total dissolution rate.

1.1.2 Solvent Representation: Fluid Simulation Methods

In the problem of object dissolving in fluid, the shape of the solid solute is continuously changing. To deal with the irregular shape of object coupling with fluid during dissolution, particle–based methods seem to be the natural choice. A unified particle–based method is used or our dissolution model which has several advantages:

- They are easy to implement with arbitrary boundaries and applied forces. Different types of particles can be mixed freely. There is no need of special handling for the solid-fluid or fluid-air interface detection.
- The computational effort is concentrated in the area where the particle density is present, so time is not wasted calculating empty areas unlike grid based methods.
- The physical and chemical processes are straightforward to add in, maintaining the extensibility to other natural phenomenon simulation, such as erosion and melting.
- There are no constraints imposed either on the geometry or in how far it may evolve from the initial conditions, such that the initial conditions can be easily programmed without the need of

complicated grid algorithms.

- It is well suited for parallelization on a GPU to achieve real-time simulation.

Smoothed Particle Hydrodynamics (SPH) is used for our solvent simulation, although other fluid particle representations are also compatible with our framework, such as Position Based Dynamic (PBD) (Macklin and Müller 2013) or Material Point Method (MPM) (Stomakhin *et al.* 2014). SPH is a mesh-free, Lagrangian, particle method for modelling fluid flows. It has been established as one of the major concepts for fluid animations in computer graphics. While SPH initially gained popularity for non-axisymmetric astrophysics simulations, it has emerged to be a fully-fledged technique for state-of-the-art engineering applications and more recently fluid animation with versatile effects.

Our particle–based method enables real–time performance using a parallel implementation on the GPU. Plausible dissolution behaviours are demonstrated in different circumstances, such as antacid pills dissolving in water, rotating and translating solids in fluids and pouring fluids onto dissolving objects. Due to the similarity between hydraulic erosion and dissolution, our dissolution model is able to simulate erosion without any additional computational overhead, such as hydraulic erosion of non-homogeneous terrains. By replacing the excitation energy with a temperature, our model is able to simulate simplified melting effects, such as a liquefying bunny.

1.1.3 Solute Representation: Blue Noise Sampling

Though our dissolution model is independent of the sampling resolution of the solutes, the distribution of the solute particles still affects the plausibility of the dissolution simulation. Random sampling may fail to guarantee the no-penetration boundary criterion during the fluid-solid interaction. Homogeneous solute favours even distributed samples with equal density. The even distribution of solute particles also improves the smoothness of the solute shape change. Simple grid sampling for solute

distribution plausibly mimics the mass transfer of the solute. However, it introduces artefacts to the fluid-solid interface and causes serrated patterns on the solute surface, affecting the rendered results in high resolution. More importantly, it causes popping artefacts due to the lack of randomness during the particle phase transition. Therefore, a blue noise sampling is needed for our solute particle distribution which generates points that are evenly yet stochastic dispersed in the desired spatial domain.

Fourier spectrum analysis shows that the spectral energy of such sample points is largely absent in the low-frequency region ('red' in terms of the visible spectrum), while evenly spread in the high-frequency region (blue). Thus, it is so-called 'blue noise'. The concept of *blue noise* was originally introduced in Ulichney (1988) — high frequency white noise. However, this definition is purely empirical. It is later analysed from a signal processing perspective by Mitchell (1991) and concluded in Heck *et al.* (2013) which can be directly applied to sampling applications. It is blue noise if:

1. the power spectrum of the sample points is noisy and without concentrated spikes, and

2. the spectrum is close to zero for low frequencies.

Blue noise sampling is a well-known technique which is useful in many graphics applications, such as image synthesis, physically-based simulation, non-photorealistic rendering, and geometry processing. Fourier spectrum analysis shows that the spectral energy of such sample points is largely absent in the low-frequency region, while evenly spread in the high-frequency region. This blue noise property turns out to greatly benefit anti-aliasing, and the sample patterns are also visually pleasing, leading to its popularity in the aforementioned applications.

There are significant prior works on blue noise sampling, such as dart throwing (Cook 1986), Lloyd relaxation (Lloyd 1982), Poisson disk sampling (McCool and Fiume 1992) and so on. One limitation of classic blue noise methods is that they are lack of controllability or difficult to vary

the distribution properties of the generated samples, which is meaningful since different applications may call for different sample patterns — those that work well for one application may not work well for others. For example, samples generated using dart throwing have a characteristic blue noise profile, but it is unclear how to modify or adapt the method to generate samples with a different profile. Recently, several techniques have been proposed to generate samples with user-specific spectral profiles (Zhou *et al.* 2012; Öztireli and Gross 2012; Wachtel *et al.* 2014). However, they are typically slow to compute and difficult to extend to general sampling domains such as surfaces and volumes.

In this thesis a novel sampling method is proposed using the well-known fluid simulation method, SPH, which solves the problems above to generate a blue noise representation in a specified region. It computes particle distributions to minimize the internal pressure variance. Experiments show that this results in sample points with a high-quality blue noise spectrum. Our method can generate a variety of blue noise samples with different distribution properties, ranging from Lloyds relaxation to Capacity Constrained Voronoi Tessellations (CCVT), and even beyond these. It is achieved by varying just one single parameter which maintains the simplicity of the sampling method. It also supports adaptive sampling and multi-class sampling with applications such as image stippling and re-meshing. Image stippling places small dots of ink onto image such that their density give the impression of the tone of the image. The stipples must be placed evenly yet randomly to avoid spurious patterns (Secord 2002). More important, our method is extremely fast and entitles the flexibility to extend to general dimensions.

The SPH sampling with CCVT profile is applied for the solute particle distribution in our dissolution simulation. Compared with other blue noise sampling methods, e.g., Poisson disk sampling, our SPH sampling provides a more evenly distributed density within the solute, which also gives the user control of the volume change happened during the solute phase transition. Our SPH sampling method for solute representation is proven to achieve better plausibility of dissolution behaviours.

1.2 Contributions of Thesis

Overall, the key contributions of this dissolution model include:

1. a unified particle framework which generates **physically and chemically plausible dissolution behaviour** that is simple to incorporate with an existing particle–based fluid system;
2. an expression for measuring the local particle excitation derived from kinetic energy which is **independent of solute sampling resolution**;
3. **a novel blue noise sampling method** for solute particle distribution which avoids solvent particle penetration and gives more control over the dissolution rate; and
4. **a mapping between the global dissolution time and local particle excitation** based on the activation energy.

The SPH sampling itself has contributions outside the application of dissolution simulation.

1. a novel blue noise sampling method inspired by SPH. Our method is **fast, easy to parallelize**, and can **produce samples with a variety of different blue noise profiles**, ranging from Lloyds relaxation to CCVT samples;
2. a sampling method **supports general domain sampling** (include curve, surface and volume), as well as **adaptive sampling** and **multi-class sampling**; and
3. a framework enables **a continuous trade-off between the Nyquist frequency and oscillation**. The relationship between the SPH kernel and its influence on the distribution properties of samples id also evaluated through experimentations.

There are other related contributions include:

1. a GPU-based region growing method used in 3D simulation;
2. a correction force for boundary problem of fluid simulation;

3. hydraulic erosion simulation of non-homogeneous terrains;

4. plausible melting simulation using SPH sampling for both solid solute and fluid solvent.

1.3 Thesis Outline

This chapter presents the motivation of dissolution simulation and SPH sampling, point out the limitations of current dissolution models and sampling methods, and specify the contributions of this work. The following chapters of the thesis are organized as below:

Chapter 2 sums up the related work of fluid simulation and describes the theoretical foundation of SPH. Chapter 3 demonstrates a GPU paralleled SPH fluid solver focused on the neighbour search algorithm. The theory builds up the foundation of the contributions presented in the subsequent parts.

Chapters 4, 5 and 6 propose a novel energy-based dissolution framework derived from the collision theory. The method is proven to be independent of sampling resolutions by comparing the examples of the same solute that is sampled with different number of particles. Our dissolution model is evaluated by demonstrating examples such as dissolving pills and hydraulic erosions. Object separation during dissolution is fulfilled using a parallel region growing method.

Chapters 7, 8 and 9 propose a new algorithm for blue noise sampling derived from SPH method. The differences between the sampling method and traditional SPH fluid solver are clarified. The sampling patterns can be controlled by tuning the fluid parameters. Different applications of SPH sampling such as image stippling and remeshing are demonstrated in chapter 10. SPH sampling is subsequently used to sample the solute in the dissolution model to achieve better plausibility of dissolution simulation in Chapter 11.

Chapter 12 concludes the thesis with a summary and suggests possible future work on the improvement of dissolution model, further investiga-

tion of sampling method such as controlling the sampling patterns by changing the kernel functions, and the potential research on medical applications such as blood clotting simulation.

Chapter 2

SPH Theory

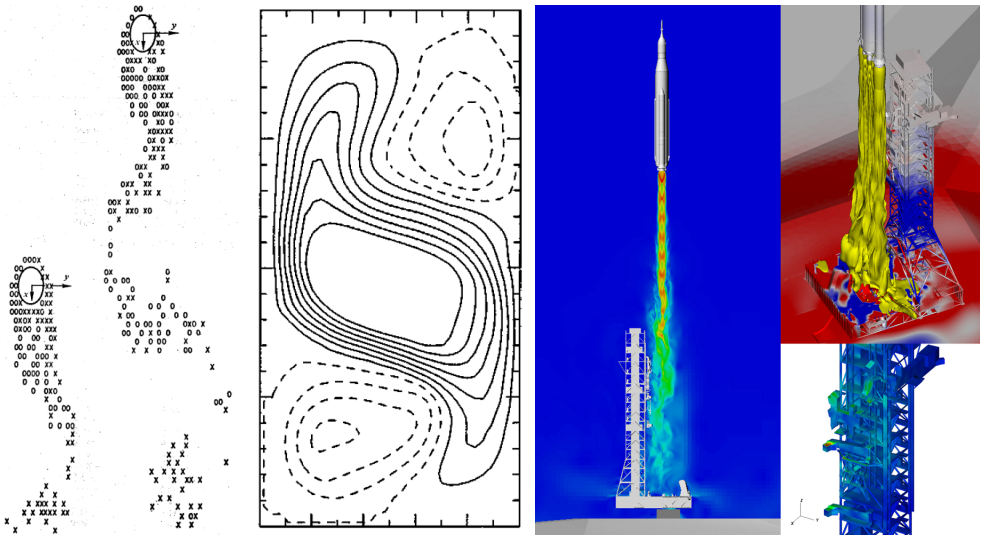
SPH is a numerical interpolation method that will be used throughout this thesis, particularly for discretizing the fluid equations of motion. Therefore, this chapter presents the physical and numerical foundation of an SPH-based fluid solver. Sec. 2.1 gives an overview of fluid simulation techniques employed in Computational Fluid Dynamics (CFD) and Computer Graphics (CG), with a particular interest in SPH. Sec. 2.2 introduces the underlying equations for an incompressible fluid, in particular the momentum equation. Subsequently, Sec. 2.3 explains how to use SPH for interpolation of fluid quantities and spatial derivatives. Finally, a GPU-accelerated SPH solver, focused on neighbour searching, is discussed in Sec. 3.1 and fluid rendering pipeline is described in Sec. 3.2.

2.1 Related Work

2.1.1 Fluid Simulations in CFD and CG

Computational Fluid Dynamics (CFD) is a very popular and important topic in fluid mechanical which has a long history since the formulation of the famous Navier-Stokes equations by Claude-Louis Navier in 1822 and George Gabriel Stokes in 1845 (Temam 2001). CFD uses numerical analysis and algorithms to solve and analyse problems that involve fluid

flows, such as finite volume methods, finite element methods, finite difference methods, boundary element methods, vorticity based methods and spectral methods (Anderson and Wendt 1995). There are a large number of methods for fluid simulations in the CFD literature (Rosenhead 1931; Chorin 1973; Leonard 1980; Hernquist 1993; Anderson and Wendt 1995; Koumoutsakos 1997; Chen and Xu 1998; Cummins and Rudman 1999). With continuous improvements and modifications, the accuracy, stability and reliability of the fluid simulation techniques are reaching an acceptable level for practical engineering applications, such as airflow simulation (Chen *et al.* 2009), plane jets test (Posner *et al.* 2003; Aristodemo *et al.* 2015), design of ejector refrigeration systems (Pianthong *et al.* 2007), analysis of tsunami wave structure (Cunningham *et al.* 2014), simulation of gas flow (Hontanon *et al.* 2000) and fuel sloshing (Longshaw and Rogers 2015).

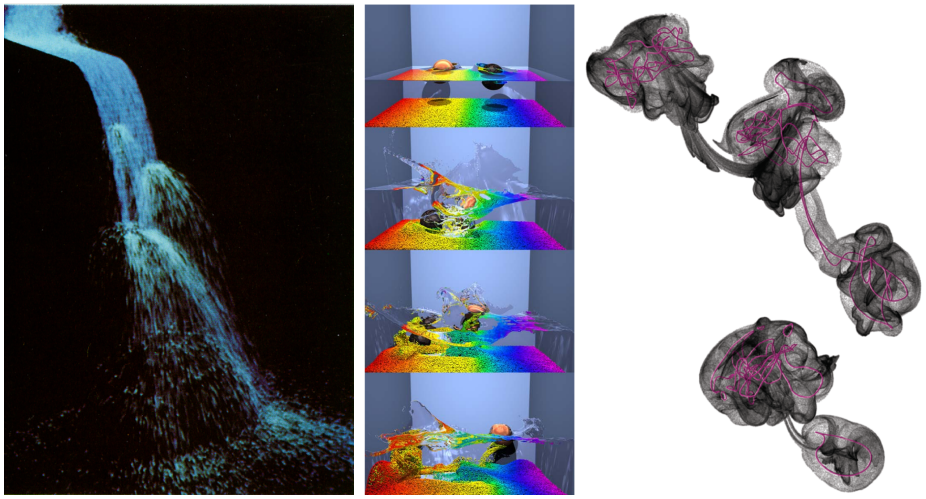


(a) (b) (c)

Figure 2.1: *Fluid simulation in CFD. (a). Flow past a circular cylinder (Chorin 1973). (b). Streamline using WCSPH (Cummins and Rudman 1999)). (c). CFD simulation of space launch system (NASA 2015).*

Compared to CFD used in engineering, where the main goal is to achieve physical accuracy, the focus of simulating fluid flow in the field of Computer Graphics (CG) is to achieve visual plausibility, efficiency and stability. Applications of fluid simulation in CG are ranging from inter-

active training environments to advertisement, game and feature film. In interactive applications such as trainings or games, a pre-defined update rate is required to guarantee a satisfying user experience. For commercials and films, on one hand the fluid animation should often be indistinguishable from reality and, thus high-resolution fluids are required to capture all prominent effects; on the other hand, the animator should be in high control in order to guide the animation to the desired result, such as ocean wave simulations with big waves on certain directions or erosion at pre-defined areas. It is difficult to force a particular fluid motion using CFD methods, unless it is a natural consequence of the system (Foster and Metaxas 1996). The fluid simulation methods in CG sacrifice the accuracy to improve the speed and control while ensuring a faithful visual approximation. This thesis will focus on reproducing fluids in the area of CG.



(a) (b) (c)

Figure 2.2: *Fluid simulation in CG. (a). Simulation of waterfall (Sims 1990), (b). Rigid fluid (Carlson* et al. *2004), (c). Smoke ring from smoke (Weißmann* et al. *2014).*

For the different purpose of fluid simulation in the fields of CG, a broad variety of techniques have been developed (see surveys in Bridson (2008) and Ihmsen *et al.* (2014b)). The pioneering work of Reeves (1983) used particle systems to model a class of fuzzy objects, such as fire and fireworks. Since then, the particle-based Lagrangian approach and the

grid-based Eulerian approach have been used to simulate fluids (Müller *et al.* 2004). Although some of the fluids are based on Navier-Stokes equations, they sometimes neglect the physical accuracy required for CFD studies, in terms of incompressibility (Carlson *et al.* 2004; Foster and Metaxas 1996), grids resolutions (Fedkiw *et al.* 2001), conservation of geometrical constraints (Elcott *et al.* 2007) and numerical viscosity (Mullen *et al.* 2009). When visual effects matter most, more elaborate CFD methods are generally considered unnecessary (Mullen *et al.* 2009).

Koumoutsakos *et al.* (2008) pointed that particle methods bridged the flow simulation in CG and CFD. An example is the fluid simulation method — SPH which is used in both CFD and CG. In CFD, SPH considers more physical properties than SPH in CG, such as the speed of sound wave, vortex spin-down, Rayleigh-Taylor instability (Cummins and Rudman 1999) and Taylor-Green problem (Xu *et al.* 2009). However, recently with the increasing computer power, SPH methods in CG adapt more techniques from CFD to enforce the incompressibility and achieve more accuracy, such as weakly compressible SPH (WCSPH)(Becker and Teschner 2007), predictive-corrective incompressible SPH (PCISPH) (Solenthaler and Pajarola 2009), local Poisson SPH (LPSPH) (He *et al.* 2012) and implicit incompressible SPH (IISPH) (Ihmsen *et al.* 2014a). While the domains of CFD and CG focus on different applications, in recent work there has been convergence, with both utilizing equivalent technical features such as parallel algorithms (Crespo *et al.* 2011).

2.1.2 Fluid Simulation in CG

The first water wave animations were presented by Fournier and Reeves (1986) and Peachey (1986). They modelled ocean surfaces using a single-valued height function which approximated ocean surfaces on a coarse scale. It was fast to compute. However, they ignored the hydrodynamics including turbulence, multi-phase and the effect of flow quantities such as pressure, leading to the difficulties of simulating many prominent effects found in fluid.

More realistic fluid simulations are later achieved by employing a physical foundation — Navier-Stokes equations. Foster and Metaxas (1996) proposed a finite-difference solution for the full set of Navier-Stokes equations and introduced the first 3D fluid solver to CG. It was based on the Marker-And-Cell (MAC) method (Harlow and Welch 1965), in which the computational domain was divided into Cartesian grids. According to the grid structure, the derivatives in the pressure solve could be easily computed using finite differences. The velocity and pressure components yielded higher accuracy than regular grids. Since then, many enhancements and variations have been proposed. An important one is semi-Lagrangian advection introduced by Stam (1999) who replaced the forward Euler integration with a backward particle trace. Thereby, the trajectory of hypothetical particles was used to advect the grid values. However, the repetitive averaging of the advected quantity blurred sharp features such as small vortices. This undesired effect, referred as *numerical dissipation*, decreased with smaller time steps.

A prominent strategy to achieve non-dissipative advection is to transport the fluid properties as done in the Fluid-Implicit-Particle (FLIP) method. The FLIP method has been first described by Brackbill and Ruppel (1986) and an adapted version has been introduced to CG by Zhu and Bridson (2005). In FLIP, the fluid is represented with particles, while the projection step is performed on an auxiliary Eulerian grid which is typically coarser sampled than the particle resolution. This method advects the particles after transferring velocities to and from the background grid, thus prevents the particles from directly affecting each other. This technique is very popular in CG (Mercier *et al.* 2015) and is still at the heart of many fluid solvers, such as *Nodes* — a commercial fluid solver of RealFlow (NEXT LIMIT 2014). The main strength of FLIP is that it is quite fast and also keeps the stabilization of the system. However, as commonly employed in CG, FLIP particles only track sub-grid velocities and are not associated with a volume, which might cause volume compression due to accumulation of numerical errors. The particles can clump or have voids, also making it difficult to create a surface for rendering.

2.1.3 SPH Methods

Another strategy to circumvent the challenges imposed by the grid discretization is to work exclusively on the particles. The most prominent representative of these mesh-free Lagrangian methods is SPH, originally developed by Gingold and Monaghan (1977) and Lucy (1977) for computational astrophysical problems. It has many attractive features include mass-conservation, Lagrangian discretization and conceptual simplicity (Macklin and Müller 2013). SPH is not a specific solver, but an interpolation method for particle systems which are employed to compute field quantities and their spatial derivatives (Müller *et al.* 2003). Since the collective movement of those particles is similar to the movement of a fluid so it can be modelled by the governing equations of the classical Newtonian hydrodynamics. The numerical method has been shown to be robust and applicable to a wide variety of fields.

SPH was first applied to free–surface flows by Monaghan (1994), and it adapted very well to the fluid simulation with complex and free–surfaces. Since then, SPH has emerged to be a state-of-the-art technique for fluid simulation with versatile effects, including gaseous phenomena and fire simulation (Stam and Fiume 1995), highly deformable bodies (Desbrun and Gascuel 1996) and lava flows (Stora *et al.* 1999). Müller *et al.* (2003) popularized the method by capturing the dynamic splashing effects of fluid in an interactive manner. It also laid the foundations for particle based approaches to interact with different substance such as fluid-fluid coupling (Müller *et al.* 2005; Losasso *et al.* 2006; Solenthaler and Pajarola 2008), fluid-solid coupling (Müller *et al.* 2004; Keiser *et al.* 2005; Solenthaler *et al.* 2007; Becker *et al.* 2009), and fluid-gas coupling (Ihmsen *et al.* 2012).

During the interaction with other substances, there is another attractive research problem — boundary conditions of the SPH method. Boundary problem is also one of the grand challenges identified by the SPH European Research Interest Community (SPHERIC) (SPHERIC 2016). The boundary problems can be classified as: solid boundaries (fluid particles penetration into boundaries of solid must be avoided);

free surface (fluid particles behaviour more compact near the fluid-air interface); inlet/outlet; initial conditions and coupling with other models (multiple phase fluid simulations). There are various techniques to solve the techniques depending on the type of conditions considered, such as Macia *et al.* (2012), Yang *et al.* (2012) and Marrone *et al.* (2013) in CFD, as well as Solenthaler and Pajarola (2008), Schechter and Bridson (2012) and Akinci *et al.* (2012) in CG.

In fluid simulation, enforcing incompressibility has always been a crucial problem. This is important in order to obtain convincing simulations of water. Standard SPH (Monaghan 1992) used a low stiffness equation of state for pressure calculation, originating in ideal gas theory. WCSPH (Becker and Teschner 2007) provided a more precise pressure estimation and required stiff equations which limited the time-step size. PCISPH (Solenthaler and Pajarola 2009) relaxed this time-step restriction by using an iterative Jacobi-style method to accumulate pressure changes. LPSPH (He *et al.* 2012) accumulated position and velocity instead of pressure to ensure the incompressibility. IISPH (Ihmsen *et al.* 2014a) proposed Pressure Poisson Equation (PPE) to compute pressure which tolerated significantly larger time steps, but at a higher cost per time-step.

There are hybrid methods which combine different components to a new solver and it is justified by benefits of one solver that addresses drawbacks of the other solver in certain scenarios and vice versa for other scenarios. A popular approach is the combination of SPH methods with grid solvers, e.g., Losasso *et al.* (2008); Lee *et al.* (2009); Raveendran *et al.* (2011); Cornelis *et al.* (2014). They inherited both the advantage of particle-based method (mass preservation) and grid-based method (efficiency and memory consumption) (Zheng *et al.* 2015).

Other approaches combine multiple resolutions. Adams *et al.* (2007) proposed adaptive particle sizes for fluid simulation. The particle size was based on geometric local feature size. It allowed more particles in geometrically complex regions. However, this method may fail when there are a lot of splashes inside fluid. Solenthaler and Gross (2011) divided

the simulation space into two parts, and allocated more computational resources to the region where complex flows emerged.

2.1.4 SPH on a GPU

SPH algorithm often becomes unstable if particles do not have enough neighbours for accurate density estimates. To avoid these situations the typical solution is taking small time steps or using sufficiently many particles. It increases the computational cost largely in both ways. Fortunately, the growth of the computational power of **Graphics Processor Unit** (GPU) (Fernando 2004) is tremendous, and the graphics cards with the latest GPUs are more affordable than in the past. The computational power of the GPU makes it possible to simulate large systems of interacting particles at interactive rates. Researchers have contributed extensively in the literature to accelerate the simulation. Some of first implementations of SPH on the GPU were Hegeman *et al.* (2006); Harada *et al.* (2007a) and Zhang *et al.* (2007). These were before the introduction of **Compute Unified Device Architecture** (CUDA) and thus imposed severe limits on the implementations. CUDA (NVIDIA 2016) is NVIDIA's parallel architecture and API (Application Programming Interface) model for GPU computing using extensions to the C/C++ language. It enables dramatic increases in computing performance by harnessing the power of the GPU. Since the appearance of CUDA, there has been growing interest in the implementation of SPH on the GPU which takes full advantage of GPU and CUDA (Ryoo *et al.* 2008; Hérault *et al.* 2010; Krog and Elster 2012; Huang *et al.* 2013; Mokos *et al.* 2015). *DualSPHysics* (Crespo *et al.* 2015), an open source project to solve free–surface flow problems, released their parallel SPH code in early 2011. GPU SPH (Hoetzlein 2014) is capable of efficiently simulating up to 8 million particles, which is very promising for use in games and films.

2.1.5 Other Fluid Simulation Methods

Since enforcing incompressibility is crucial but computationally expensive, new fluid solvers are proposed such as position based method (PBD) (Müller *et al.* 2007; Macklin and Müller 2013), which operates on the positions of particles directly via constraints and convergences similar to SPH solvers. Borrowing the idea of a density estimator from SPH, fluid incompressibility is enforced with a density constraint (Bodin *et al.* 2012). Macklin *et al.* (2014) established a PBD framework to simulate a wide range of physical phenomena. Most recently, it is used to simulate extraction and liquids mixing (Yang *et al.* 2015).

Stomakhin *et al.* (2014) introduced a novel Material Point Method (MPM) for phase transition and varied materials, such as melting. It can be easily augmented with a Chorin-style projection technique (Chorin 1968) that enables simulation of arbitrarily incompressible materials thus providing a connection to the FLIP techniques. Yue *et al.* (2015) applied MPM for dense foam simulation.

de Goes *et al.* (2015) proposed the idea of Power Particle which considered the fluid particles as non-overlapping volumetric parcels rather than material points. These 'power' particles precisely controlled particle density and pressure without the need of kernel estimations or artificial viscosity.

2.2 Fluid Dynamics

Various models have been proposed to approximate the behaviour of liquids. Most of fluid simulation, include SPH, is governed by the famous *Navier-Stokes equations*.

2.2.1 The Navier-Stokes Equation

Navier-Stokes equations arise from Newton's second law and describe the motion of viscous fluid substances (Temam 2001). They are of great

interest in scientific and engineering senses. They cannot only be used to model the fluid flow (Marshall *et al.* 1997; Stam 1999) but also optimize the design of car and aircraft (Zhu and Kronast 1993; Eyi *et al.* 1994; Kroll *et al.* 2002; Krajnovic and Davidson 2004). Surprisingly, they are also useful in solving financial problem (Jensen *et al.* 2003), analysing the stock market (Zocco 2007), and even predicting electricity spot prices (Shcherbacheva 2011). Another interesting problem about Navier-Stokes equations are themselves. Despite the wide range of practical uses, Navier-Stokes equations have not yet been proven to be always exist in 3D solution or smooth (do not contain any mathematical singularity) which is called the *Navier-Stokes existence and smoothness problems* (Clay Mathematics Institute 2016).

The Navier-Stokes equations are a set of partial differential equations which describe the conservation of motion of the fluid. The equations are usually written as (Bridson 2008):

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} + \frac{1}{\rho} \nabla p = \mathbf{g} + \mu \nabla \cdot \nabla \mathbf{v} \tag{2.1}$$

$$\nabla \cdot \mathbf{v} = 0 \tag{2.2}$$

where $\mathbf{v}$ is the velocity of the fluid, ρ and p separately stand for the density and pressure of the fluid, $\mathbf{g}$ is the acceleration due to gravity which can be extended to all additional control forces, μ is the viscosity coefficient which measures how much the fluid resists deforming while it flows (or move intuitively, how difficult it is to stir).

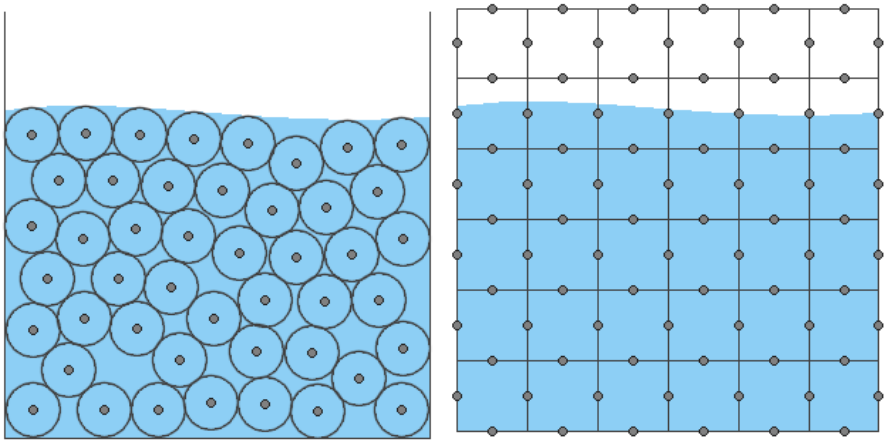
The two functions control the fluids in two ways: the conservation of momentum (Eq. 2.1) describes the movement and evolution of the fluid; and the differential is the incompressibility condition which shows the conservation of mass (Eq. 2.2).

2.2.2 The Material Derivative

There are two viewpoints to describe a continuous volume: the *Lagrangian* methods and the *Eulerian* methods (Bridson 2008).

Lagrangian approach represents the volume with a discrete set of particles. The material is described by watching the trajectory of each individual particle. All the computational information is carried by the particles which are free to move (Premože *et al.* 2003). Fig. 2.3(a) shows a basic layout of a 2D particle-based fluid.

Eulerian approach calculates the material quantities at fixed points in space, such as the velocity and temperature, instead of tracking each particle. The Eulerian approach corresponds to using a fixed grid that does not change with the fluid flow. Fig. 2.3(b) depicts a basic layout of a 2D grid-based fluid.



(a) Lagrangian approach (b) Eulerian approach

Figure 2.3: *Fluid diagrams in 2D (Kelager 2006). (a). Lagrangian particle-based method. The particles are presented by the dots with the circles plotting out the volume of each particle. (b). Eulerian grid-based method. The dotes show the discrete quantity field.*

The *material derivative* serves as a key link between the Eulerian and Lagrangian descriptions of continuum deformation. The material derivative $\frac{D}{Dt}$ is defined for any tensor field that is macroscopic, with the sense that it depends only on position and time coordinates. For

velocity, the material derivative is linked to the Eulerian derivative:

$$\frac{D\mathbf{v}}{Dt} = \frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} \tag{2.3}$$

The first term of material derivative $\frac{\partial \mathbf{v}}{\partial t}$ shows how fast the velocity is changing at a fixed point in space. The second term, $\mathbf{v} \cdot \nabla \mathbf{v}$ is correcting for how much of that change is due just to the differences in the fluid flowing past. Expand Eq. 2.3 in Cartesian form using the partial derivatives with $\mathbf{x} = (x, y, z)$ and $\mathbf{v} = (u, v, w)$. It can be written as:

$$\frac{D\mathbf{v}}{Dt} = \frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \frac{\partial \mathbf{v}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z} \end{bmatrix} \tag{2.4}$$

In the Lagrangian method, the particles move with the fluid and carry the attributes of the fluid. Thus the convection derivative $\mathbf{v} \cdot \nabla \mathbf{v}$ is not needed for the particle systems, such as SPH (Müller *et al.* 2003). Although this thesis only focuses on the Lagrangian approach SPH, the material derivative is employed in the following to derive the equation of motion for fluids. Thereby, a general description is obtained that can be mapped onto both the Lagrangian and Eulerian method.

Lagrangian approach vs. Eulerian approach

Compared to the Lagrangian method, the Eulerian method is more suitable for fluids within a fixed boundary and is easier to deal with the spatial derivatives such as the pressure gradient and viscosity. A major disadvantage is the grid itself. The fluid is constrained to stay within the grid. It simply cannot exist outside the grid domain, and thus makes it a difficult problem to let the fluid flow naturally, e.g., when a fluid container fractures.

The Lagrangian approach on the other hand is not necessarily constrained to a finite box, thus is easier to implement with arbitrary boundaries and applied forces. It is ideal for simulating interaction with arbitrary rigid bodies and free–surfaces (Bridson and Müller-Fischer 2007).

The method only performs computation where necessary and requires less storage and bandwidth since the model properties are only stored at the particle positions, rather than at every point in space. In addition, particles can easily be generated and deleted dynamically as needed (Müller *et al.* 2005). The main disadvantage of the Lagrangian approach is the high computational cost for the large number of particles. Thanks to the modern GPUs with massive parallel computation capabilities, it is possible to simulate such huge number of particles at interactive rates. The particles can directly be used to render the fluids surface.

With the Lagrangian description of motion, the material derivatives includes the non–linear advection terms which in turn avoids needing to use very expensive and complicated CFD algorithms such as upwinding (Anderson and Wendt 1995).

2.2.3 The Momentum Equation

To derive the motion equation for fluids, first consider the continuous fluid volume as a set of parcels. These parcels cannot be considered as point masses, but as volumetric elements. Each parcel represents a piece of volume V with mass m. Thus, to derive the equation of motion for fluids, it is more appropriate to consider the momentum change of the volumes. To integrate the system forward in time all we need is to identify what the forces acting on each parcel are, and then use Newton's second law $\mathbf{F} = m\mathbf{a}$ to calculate the acceleration. Assuming constant property for each parcel, this can be expressed as:

$$\mathbf{F} = m\frac{D\mathbf{v}}{Dt} = \rho V\frac{D\mathbf{v}}{Dt} \tag{2.5}$$

where the mass m is defined as volume V times density ρ. However, for infinitesimal volumes ($V \to 0$), the net force on the element would go to zero. Therefore, in fluid mechanics, the force per unit volume, referred as *force density*, is of more relevance than the force. It is denoted by a

lower case $\mathbf{f}$ and defined as

$$\mathbf{f} = \frac{\mathbf{F}}{V} = \frac{\rho D\mathbf{v}}{Dt} \tag{2.6}$$

which states that the force per unit volume equals the time-rate of change of momentum per unit volume. The acceleration can be calculated as:

$$\mathbf{a} = \frac{\mathbf{f}}{\rho} = \frac{D\mathbf{v}}{Dt} \tag{2.7}$$

If the constant property assumption is removed, the force and temporal change of momentum have to integrate over the volume region $V(t)$ which yields

$$\int_V \mathbf{f}\mathrm{d}V = \int_V \frac{\rho D\mathbf{v}}{Dt}\mathrm{d}V \tag{2.8}$$

To integrate the system forward in time, the total force needs to be determined. Basically, there are two kinds of forces acting on fluids, *body forces* $\mathbf{f}^B$ and *surface forces* $\mathbf{f}^S$.

Body forces are long-range forces acting on the entire volume region V. Their origin is far away and their strength varies very slowly. Accordingly, they act uniformly on all particles. A typical body force is the gravity:

$$\mathbf{f}^g = \frac{m\mathbf{g}}{V} = \rho\mathbf{g} \tag{2.9}$$

Surface forces are short-range forces acting only on the surface S of the entire volume. Plugging the partial forces into Eq. 2.8, it becomes:

$$\int_V \frac{\rho D\mathbf{v}}{Dt}\mathrm{d}V = \int_V \mathbf{f}^B\mathrm{d}V + \int_S \mathbf{f}^S\mathrm{d}S \tag{2.10}$$

The surface forces are imposed by surrounding fluid elements due to the normal stress distributions which are acting in normal direction $\mathbf{n}$ and the shear stresses which are acting tangentially to the surface. $\mathbf{n}$ is a unit vector pointing outwards from the surface. The net surface force can thus be written as $\mathbf{f}^S = \boldsymbol{\sigma} \cdot \mathbf{n}$, where $\boldsymbol{\sigma}$ is a 3×3 matrix, which is the stress tensor of $\mathbf{f}^S$. The surface integral can be converted to a volume

integral by applying the Gauss theorem (Whittaker 1935):

$$\int_S \mathbf{f}^S \mathrm{d}S = \int_S \boldsymbol{\sigma} \cdot \mathbf{n} \mathrm{d}S = \int_V \nabla \cdot \boldsymbol{\sigma} \mathrm{d}V \tag{2.11}$$

By shrinking the volume of the particle to an infinitesimal value, the integro-differential form of Eq. 2.10 can be transformed into a pure differential form:

$$\rho \frac{D\mathbf{v}}{Dt} = \mathbf{f}^B + \nabla \cdot \boldsymbol{\sigma} \tag{2.12}$$

The surface force can be decomposed into two components: pressure and viscosity. **Pressure** is a scalar that is independent of orientation, i.e. it is an isotropic second-order tensor. As long as pressure does not vary spatially, the net force due to pressure is zero. However, if there is an imbalance, higher pressure on one side than on the other side, resulting in a force pointing from high pressure regions towards low pressure regions. By considering the conservation of the momentum on a fluid element at a point in space, the pressure gradient terms can be calculated in the form of $-\nabla p$. **Viscosity** describes the resistance of the fluid to deform. It tries to make the particle move at the average velocity of the nearby particles and has the physical effect of minimizing local velocity differences. Viscosity can be described as internal friction and describes the diffusion of momentum. By considering the conservation of the momentum on a fluid element at a point in space, the Laplacian operator terms $\nabla \cdot \nabla$ represent the diffusion of momentum. Therefore, viscosity can be written as $\eta \nabla \cdot \nabla \mathbf{v}$, where η is the dynamic viscosity coefficient.

Consequently, the Eq. 2.12 can be summarised as the **momentum equation** Eq. 2.2 which really is Newton's equation $\mathbf{F} = m\mathbf{a}$ in disguise. It shows how the fluid accelerates due to forces acting on it.

$$\rho \frac{D\mathbf{v}}{Dt} = -\nabla p + \mu \nabla \cdot \nabla \mathbf{v} + \rho \mathbf{g} \tag{2.13}$$

where $\mu = \eta / \rho$ denotes the kinematic viscosity coefficient which defines how strongly the viscosity force acts. Fluids like honey have high

viscosity, and fluids like water have low viscosity.

2.2.4 Incompressibility

In animation fluids are generally considered to be incompressible, which means their volume does not change. The rate of volume change of the fluid is estimated by integrating the normal component of its velocity around the surface boundary. For incompressible fluid, the volume stays constant and the rate of change is zero.

$$\int_S \mathbf{v} \cdot \mathbf{n} \mathrm{d}S = \int_V \nabla \cdot \mathbf{v} \mathrm{d}V = 0 \tag{2.14}$$

The equation should be true for any region of fluid. Independent of the volume integration, the only continuous function that integrates to zero is zero itself. Thus the integrand has to be zero everywhere:

$$\nabla \cdot \mathbf{v} = 0 \tag{2.15}$$

This is the **incompressibility condition** of the Navier-Stokes equation. This equation expresses that volume is conserved if and only if the velocity field is *divergence free* meaning that the divergence of the velocity field is zero.

The use of particle-based methods simplifies the Navier-Stokes. Firstly, because the number of particles during simulation is constant and each particle represents a fixed amount of mass, the conservation of mass is guaranteed (Bridson 2008). Secondly, since the particles move with the fluid, the substantial derivative of the velocity field is simply the time derivative of the velocity of the particles (Müller *et al.* 2003).

2.3 Smoothed Particle Hydrodynamics

SPH is an interpolation method which uses finite discrete particles to approximate a continuous field. The Lagrangian particles carry quantities,

e.g. mass, position, velocity, etc., and also other estimated physical field quantities dependent of the problems, e.g., mass-density, temperature, pressure, etc.

The particles move with the flow. In each time step, the physical quantities are computed by updating the fluid equations for each particle. As depicted in the previous section, the fluid dynamics is described by spatial derivatives of the quantities. Therefore, any numerical fluid solver has to approximate these derivatives. In Eulerian methods, the derivatives can be easily computed as finite differences using the grid vertices. In contrast, SPH particles are located completely arbitrarily and not spatially fixed, therefore another concept — interpolation is used for approximating the spatial derivatives.

SPH integrates the hydrodynamic equations of motion on each particle in the Lagrangian formalism. Relevant physical quantities are computed for each particle as an interpolation of the values of the nearest neighbouring particles, and then particles move according to those values. The conservation laws of continuum fluid dynamics, in the form of partial differential equations, are transformed into their particle forms by integral equations through the use of an interpolation function that gives the kernel estimate of the field variables at a point. Computationally, information is known only at discrete points (the particles), so that the integrals are evaluated as sums over neighbouring particles.

This section first explains the interpolation concept on which SPH is based and describes how the spatial derivatives can be computed, then specifies the SPH force calculations, and finally discusses the boundary conditions, time integrations and smooth kernels.

2.3.1 SPH Interpolation and Spatial Derivatives

In SPH, a finite set of volumetric particles are interpolated to evaluate the fluid quantities at any position within the fluid space. To do that, SPH employs the concept of integral representation of field variables. Any field variable $A(\mathbf{x})$ at position $\mathbf{x}$ defined in a domain Φ can be

expressed by:

$$A(\mathbf{x}) = \int_{\Phi} A(\mathbf{x}_j) W(\mathbf{x} - \mathbf{x}_j, h) d\mathbf{x}_j, \quad \forall \mathbf{x}_j \in \Phi \tag{2.16}$$

where $\mathbf{x}_j$ is any point in Φ and W is radially symmetric smoothing kernels, discussed further in Sec. 2.3.5. The numerical approximation of discretisation to Eq. 2.16 can be obtained by a weighted sum of contributions from neighboring particles j in Φ.

$$A(\mathbf{x}) = \sum_j V_j A(\mathbf{x}_j) W(\mathbf{x} - \mathbf{x}_j, h), \quad \forall \mathbf{x}_j \in \Phi \tag{2.17}$$

where V_j is the volume attributed to particle j. $A(\mathbf{x}_j)$ is the value of any quantity A at location $\mathbf{x}_j$, abbreviated as A_j. h is the core radius of the kernel which controls the smoothness or roughness of the kernel, referred to as the *smoothing length*. Practically, h also defines the support of Φ around each particle. If the distance between particles is larger than h, they have no effect with each other on the calculation. With $V = m/\rho$, the equation can also be written as (Müller *et al.* 2003):

$$A(\mathbf{x}) = \sum_j \frac{m_j}{\rho_j} A(\mathbf{x}_j) W(\mathbf{x} - \mathbf{x}_j, h), \quad \forall \mathbf{x}_j \in \Phi \tag{2.18}$$

The mass m_j of particle j is constant during the simulation while the density ρ varies. The density at position $\mathbf{x}$ can be calculated as:

$$\rho(\mathbf{x}) = \sum_j m_j W(\mathbf{x} - \mathbf{x}_j, h) \tag{2.19}$$

Derivatives of the quantities are also needed in most fluid equations. They only effect the smoothing kernel in SPH equations. The gradient of A is calculated as:

$$\nabla A(\mathbf{x}) = \sum_j m_j \frac{A_j}{\rho_j} \nabla W(\mathbf{x} - \mathbf{x}_j, h) \tag{2.20}$$

and the Laplacian of A is:

$$\nabla^2 A(\mathbf{x}) = \sum_j m_j \frac{A_j}{\rho_j} \nabla^2 W(\mathbf{x} - \mathbf{x}_j, h) \tag{2.21}$$

This is a very important property for numerical fluid simulations as it allows a straightforward approximation of the spatial derivatives.

2.3.2 SPH Force Calculation

2.3.2.1 Pressure Force

Following the SPH rule of Eq. 2.20, the pressure gradient term of incompressible fluids $-\nabla p$ can be formulated as:

$$\mathbf{f}^{\mathrm{p}} = -\sum_j m_j \frac{p_j}{\rho_j} \nabla W(\mathbf{x} - \mathbf{x}_j, h) \tag{2.22}$$

However, this formulation is not symmetric when only two particles interact and does not preserve linear and angular momentum as the sum of forces is not zero in general. Furthermore, even if the pressure field is constant, $\mathbf{f}^{\mathrm{p}}$ is not necessarily zero. In this case, the force is only zero if either the pressure values are zero or the neighbourhood is sampled symmetrically. Different ways of symmetrization of Eq. 2.22 have been proposed in the literature (Müller *et al.* 2003; Monaghan 2005). This thesis uses the solution from Monaghan (2005) as below for the purpose of stability.

$$\mathbf{f}_i^{\mathrm{p}} = -\rho_i \sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W(\mathbf{x} - \mathbf{x}_j, h) \tag{2.23}$$

where p_i and p_j are separately the pressure of particles i and j at location $\mathbf{x}_i$ and $\mathbf{x}_j$. The pressure is estimated using the method of the standard SPH via a gas constant k and the rest density ρ_0 (Müller *et al.* 2003), since the strict incompressibility is not the focus of this thesis.

The pressure can be calculated as:

$$p = k(\rho - \rho_0) \tag{2.24}$$

The pressure gradient force is created whenever there is a pressure difference and it acts in the direction from high pressure to low pressure along the negative of the pressure gradient. The pressure depends on the mass-density of the fluid. A higher concentration of mass-density in a volume or a higher density gives rise to a repulsive force, while a density smaller than the rest density results in an attractive force. As such, the pressure gradient term aims to equalise the density differences throughout the fluid. The density difference $(\rho - \rho_0)$ will be proven to be important when developing an SPH sampling method in Chap. 8.

2.3.2.2 Viscosity Force

The viscosity term $\mu\nabla\cdot\nabla\mathbf{v}$ again needs to yield asymmetric rules because the velocity field varies from particle to particle. Since viscosity forces are only dependent on velocity differences and not on absolute velocities, there is a nature way to symmetrize the viscosity forces by using the velocity differences:

$$\mathbf{f}^{\mathrm{v}} = \mu \sum_j m_j \frac{\mathbf{v}_j - \mathbf{v}_i}{\rho_j} \nabla^2 W(\mathbf{x} - \mathbf{x}_j, h) \tag{2.25}$$

If a standard Gaussian kernel is used for viscosity, the use of $\nabla^2 W$ can result in negative force when two particles get close to each other which will then increase the relative velocity. Therefore in CFD normally two times of ∇W are used instead of $\nabla^2 W$ (Crespo *et al.* 2015). Here, in CG since the accuracy is not the primary of fluid simulation, an alternative kernel is proposed to solve the problem. More details will be discussed in Sec. 2.3.5

Viscosity force provides the resistance to the flow of particles, hence it defines how viscous the liquid is, such as in the case of water against

honey. The Laplacian of the velocity field gives the divergence from its average value and thus, this force aims to restore or smooth the velocity difference.

2.3.2.3 Surface Tension Force

The surface tension force is an external force acts only on particles at the surface of the fluid. It is normally not a part of the Navier-Stokes equations and is more considered as a boundary condition.

Molecules inside a fluid are held in perfect balance through intermolecular forces that are equal in all directions. However, this is not the case for particles at the surface of the fluid and this imbalance gives rise to the surface tensions. To bind the fluid surface together, the surface tension force acts in the direction of the inward surface normals towards the fluid. The surface tension force aims to minimize the curvature of the surface and thus, smooths the fluid surface (Müller *et al.* 2003). Fig. 2.4 depicts the behaviour of the surface tension forces.

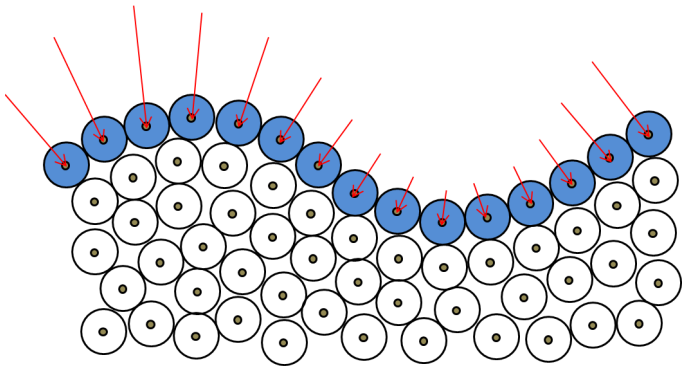


Figure 2.4: *Behaviour of the surface tension force. A positive curvature (left) generates a stronger surface tension force than a negative curvature (right), but the surface tension force will in all cases apply in the direction towards the fluid.*

The surface of the fluid can be identified using an additional *color field*:

$$c(\mathbf{x}) = \sum_j \frac{m_j}{\rho_j} W(\mathbf{x} - \mathbf{x}_j, h) \tag{2.26}$$

A particle is a surface particle if

$$\|\nabla c\| > l \tag{2.27}$$

where l is a threshold parameter. The direction of the surface normal at the location of $\mathbf{x}$ is pointing outwards the fluid and can be represented as

$$\mathbf{n} = -\nabla c(\mathbf{x}) \tag{2.28}$$

The curvature of the surface is calculated from the Laplacian of the color field:

$$k_c = \frac{\nabla \mathbf{n}}{\|\mathbf{n}\|} = -\frac{\nabla^2 c}{\|\mathbf{n}\|} \tag{2.29}$$

The behaviour of the surface tension force acts along the inward surface normals, in the direction towards the fluid:

$$\mathbf{f}^{\mathrm{t}} = -\sigma k \mathbf{n} = \sigma \nabla^2 c \frac{\mathbf{n}}{\|\mathbf{n}\|} \tag{2.30}$$

where σ is the surface tension coefficient, depending on the fluids which form the surface. The surface tension force is asymmetric by design. In the real world surface tension forces are the consequence of fluid-air interactions. Only a tiny fraction of the real world is modelled, and there is no air particles to symmetrize the surface tension force, thus the model might to some extent be unrealistic.

2.3.3 Boundary Conditions

To create convincing fluid simulation, a practical way is to involve environmental interaction, such as defined boundaries or obstacles to constrain the fluid motion. Special considerations have to be taken into account at boundaries in order to prevent spatial and temporal discontinuities of physical properties. Discontinuities might lead to perceivable simulation artefacts and numerical instabilities.

These thesis only focuses on two boundary conditions, **fluid-solid**

interface and **free–surface**. The boundary (or interface) between two different fluids is not considered in this work - refer to Hong and Kim (2005) for more information on this topic.

For fluid-solid interface, the fluid must not flow into or out of the solid. It is simple to phrase this in terms of velocity: the normal component of velocity at the interface has to be zero:

$$\mathbf{v} \cdot \mathbf{n} = 0 \tag{2.31}$$

where $\mathbf{n}$ is the normal of the fluid-solid interface. If the solid is moving, the normal component of the fluid velocity needs to match the normal component of the solid's velocity $\mathbf{v}_s$, so that the relative velocity has zero normal component:

$$\mathbf{v} \cdot \mathbf{n} = \mathbf{v}_s \cdot \mathbf{n} \tag{2.32}$$

This is called *no-penetration* condition. The normal component of the velocity is restricted to avoid the fluid particle into the solid, but the tangential components are completely decoupled which allows the fluid to freely slip past tangentially — *no-stick* condition. This condition is only for inviscid fluid. For a visibly viscous fluid, the kinematic viscosity coefficient μ (stickiness) might have an influence on the tangential component of the fluid's velocity. In this case, the tangential component of the velocity will enter the viscosity term which gives rise to the desired stickiness. Schechter and Bridson (2012) introduced ghost particles to solve the particle deficiency at the boundaries. Each ghost density is set to the nearest liquid particle's density which naturally enforces no-penetration and no-stick boundary conditions.

The other boundary condition that people are interested in is at free surface which really is fluid-air interface. Fluid molecules are more strongly attracted to other fluid molecules than they are to air molecules. Therefore, the fluid molecules at the interface try to move to be as surrounded by fluid as much as possible, which in a geometric perspective is to minimize the surface area. This can be interpreted as a surface tension force detailed in Sec. 2.3.2.3. Akinci *et al.* (2012) proposed a

novel surface tension model and adhesion model to solve the boundary problem without the need of ghost particles.

2.3.4 Time Integration

The acceleration of the particle is computed by dividing the net of density force by their density, as in Eq. 2.7:

$$\mathbf{a} = \frac{1}{\rho}(\mathbf{f}^{\mathrm{p}} + \mathbf{f}^{\mathrm{v}} + \mathbf{f}^{\mathrm{t}} + \mathbf{f}^{\mathrm{g}}) \tag{2.33}$$

The acceleration is then integrated to update the velocity and position of the particle. Two integration methods have been implemented in this thesis, namely semi-implicit Euler and leap-frog.

Semi-Implicit Euler is a first-order integrator, just as the standard Euler method. It is known to conserve energy. This is especially useful when computing orbital dynamics, since many other integration schemes do not conserve energy and allow the system to drift substantially over time, such as the Runge-Kutta 4th order method (University of South Florida 2016). However, the Semi-Implicit Euler method suffers from numerical instabilities with big time-steps. It updates the velocity and position by iterating:

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \mathbf{a}_n \Delta t \tag{2.34}$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{v}_{n+1} \Delta t \tag{2.35}$$

Leap-frog integration is a second order method that is more accurate than the first-order Euler method and also conserves the energy. The name leapfrog comes from one of the ways to write this algorithm, where positions and velocities 'leap over' each other. It calculates velocities and positions at interleaved times, with the velocities calculated at half times and the positions calculated at full integer times. It is stable for oscillatory motion as long as the time step Δt is constant. The

velocity and position are calculated as follows (Hut and Makino 2004):

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \frac{\mathbf{a}_n + \mathbf{a}_{n+1}}{2}\Delta t \tag{2.36}$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{v}_n \Delta t + \frac{1}{2}\mathbf{a}_n \Delta t^2 \tag{2.37}$$

The leap-frog time integration is used for the implementations of the examples in this thesis. Since the velocity is calculated from the average of the current and next acceleration, the integration provides more stability to the next displacement of the particles. It also has another interesting feature of time-reversibility. One can integrate forward n steps, and then reverse the direction of integration and integrate backwards n steps to arrive back at the same starting position. This means that in the absence of friction, the scheme is symplectic (Leimkuhler *et al.* 1996).

2.3.5 Smoothing Kernel

Using a different smoothing kernel in SPH is equivalent to using a different filter function, thus the choice of smoothing kernels for a specific problem is of significant importance. The stability, accuracy and speed of SPH method highly depend on the smoothing kernels.

The kernel must be normalized. The unit integral ensures that maxima and minima are not enhanced.

$$\int_{\Phi} W(\mathbf{r}, h) d\mathbf{r} = 1 \tag{2.38}$$

where $\mathbf{r}$ is the abbreviation of $(\mathbf{x} - \mathbf{x}_j)$ with $r = \|\mathbf{r}\|$. The kernel is also an even function, which means

$$W(\mathbf{r}, h) = W(-\mathbf{r}, h) \tag{2.39}$$

thus rotational symmetry is enforced, which is useful to ensure in-

variance under rotations of the coordinate system. With Eq. 2.38 and Eq. 2.39 obtained, the interpolation is of second order accuracy and the error of approximating Eq. 2.16 and Eq. 2.18 is $O(h^2)$ or better. In Monaghan (1992), it is also suggested that a suitable kernel should have a limited or compact support radius, in order to ensure zero kernel interactions outside the computational range of the radius, which implies

$$W(\mathbf{r}, h) = 0, \|\mathbf{r}\| > h \tag{2.40}$$

In this thesis, the kernels used are shown as below:

Poly6 Kernel (Müller *et al.* 2003) for the calculation of the density and surface tension force:

$$W_{poly6}(\mathbf{r}, h) = \frac{315}{64\pi h^9} \begin{cases} (h^2 - r^2)^3 & \|\mathbf{x}\| < h \\ 0 & \text{otherwise} \end{cases} \tag{2.41}$$

$$\nabla W_{poly6}(\mathbf{r}, h) = -\mathbf{r}\frac{945}{32\pi h^9}(h^2 - r^2)^2 \tag{2.42}$$

$$\nabla^2 W_{poly6}(\mathbf{r}, h) = -\frac{945}{32\pi h^9}(h^2 - r^2)(3h^2 - 7r^2) \tag{2.43}$$

The advantage of this kernel is that r only appears squared which avoids computing square roots for distance. However, it cannot be used for the computing of the pressure forces. Because the kernel gradient approaches zero at the centre which means the repulsion force vanishes as particles get very close, leading to clusters.

Spiky Kernel (Desbrun and Gascuel 1996) is thus proposed for pressure computations to generate the necessary repulsion forces:

$$W_{spiky}(\mathbf{r}, h) = \frac{15}{\pi h^6} \begin{cases} (h - r)^3 & \|\mathbf{x}\| < h \\ 0 & \text{otherwise} \end{cases} \tag{2.44}$$

$$\nabla W_{spiky}(\mathbf{r}, h) = -\frac{45}{\pi h^6}\frac{\mathbf{r}}{\|r\|}(h - r)^2 \tag{2.45}$$

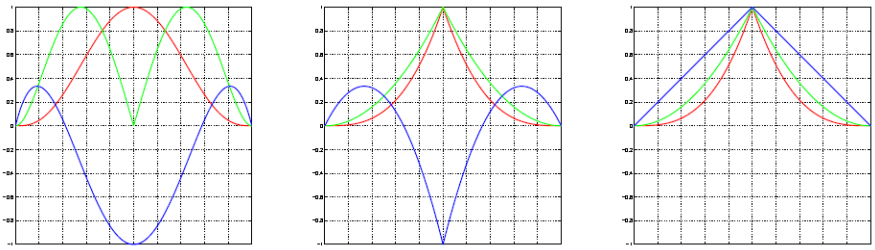


Figure 2.5: *Kernels used in SPH method. The red lines show the kernels, the green ones are the corresponding gradient kernels, and the blue ones are the corresponding Laplacian kernels.*

Viscosity is the characterisation of the diffusion of momentum. It have a smoothing effect on the velocity field. Both the Laplacian of the Poly6 and Spiky kernel can get negative when two particles get close to each other, which will result in forces that increase their relation velocity, causing stability problems in real-time application. Thus a kernel is needed whose Laplacian of the smoothed velocity field is positive everywhere. Therefore, Müller *et al.* (2003) designed the viscosity Kernel:

$$W_{viscosity}(\mathbf{r}, h) = \frac{15}{2\pi h^3} \begin{cases} -\frac{r^3}{2h^3} + \frac{r^2}{h^2} + \frac{h}{2r} - 1 & \|\mathbf{x}\| < h \\ 0 & \text{otherwise} \end{cases} \tag{2.46}$$

$$\nabla^2 W_{viscosity}(\mathbf{r}, h) = \frac{45}{\pi h^6}(h - r) \tag{2.47}$$

It is worth noting that the kernels introduced above are the kernels used in 3D examples, while the coefficient of the kernels will be different in 2D to secure the normalization criterion. These three kernels are visualised in Fig. 2.5.

2.4 Summary

This chapter reviews the fluid simulation techniques in both Computational Fluid Dynamics (CFD) and Computer Graphics (CG), presents

the physical and numerical foundation and explains the underlying equations for SPH method. Understanding the underlying theory of SPH is important fot the implementation of SPH. In the next chapter, a GPU-based SPH implementation will be discussed, as well as the rendering pipeline for fluid simulation.

Chapter 3

SPH Implementation

The main limitation of SPH is the requirement for a very large number of particles to obtain realistic results. However, the computational burden is very heavy, especially in 3D. This thesis uses CUDA (Fernando 2004), a parallel computing architecture developed by NVIDIA, to accelerate the framework for SPH. The ability of CUDA to perform scattered memory writes makes it possible to build dynamic data structures on the GPU. Since most of the SPH computation can be done on GPUs, the SPH algorithm can exploit massive computational power of GPUs. The number of particles can be considerably increased and the simulation is greatly accelerated compared to CPU-based SPH simulations. The GPU-based SPH implementation is based on the work of Hoetzlein (2014).

3.1 Neighbour Search

SPH is an interpolation of the properties of the neighbouring particles, that is the calculations of the sums over the surrounding particles. The summation term is computational intensive; it requires the list of interacting particles to be updated every time step since the particles do not have any fixed connectivity in SPH. Therefore, the computational cost of this algorithm mainly depends on the efficiency of the neighbour search.

Thus, the efficient querying and processing of particle neighbours become crucial for the performance of the simulation. The naive neighbour search over all particles would result in a computation complexity that is quadratic of the particle number $O(N^2)$. For the amount of particles aiming at it is impracticable. Therefore, it is necessary to implement an efficient acceleration structure for the neighbour search.

3.1.1 Thread Block

SPH computations are naturally suitable for parallelization. A thread is assigned to each particle in the simulation. Each thread is then responsible for calculating the SPH summation. Threads are organized in blocks. The number of threads per block should not be chosen arbitrarily (Volkov 2010). The first thing to consider is that CUDA generally requires more than 100 threads per thread block to fully hide memory latencies. Also the block size should be a multiple of the warp size (32 on the used GPU) to guarantee coalesced reading and writing and thus achieve peak memory bandwidth over the relatively slow GPU global memory channel. In addition, the limitation of the shared memory as well as the number of registers per block sets the upper limit on the number of threads inside a block (1024 on the used GPU). These values can be simply obtained from an initialization-time query to the hardware, shown in Fig. 3.1. The GPU occupancy (NVIDIA, CUDA 2010) is calculated according to the compute capability and resource usage, shown in Fig. 3.2. By considering all the factors above, 192 threads per block is assigned in our implementation which peaks the GPU occupancy.

3.1.2 Data Structure

The key idea of the parallel neighbour search is the spatial partitioning scheme. By dividing the simulation space into small cells and performing many independent operations in parallel on these cells it is easier to find the neighbours of a given particle. As reported in Harada *et al.* (2007a), the construction cost of a typical hierarchical grid is $O(N \log N)$ and the

Figure 3.1: *GPU information query.*

cost of accessing a leaf node is $O(\log N)$. In contrast, the construction cost of a uniform grid is $O(N)$, while any item can be accessed in $O(1)$. Therefore, employing uniform grids seems to be the most efficient and the consensus for standard SPH. Other acceleration structures used in static applications, e.g. kd-tree, might be too expensive to construct and is beyond the scope of this thesis. The complexity of the SPH algorithm on CPU and GPU is shown in as Table 3.1.

	Step Complexity
CPU without special grid structure	$O(N^2)$
CPU with uniform grid structure	$O(N)$
GPU with uniform grid structure	$O(1)$

Table 3.1: *Step complexity of SPH algorithm on the CPU and GPU.*

A uniform grid subdivides the simulation space into a grid of uniformly sized cubic cells. For simplicity, a grid where the cell size equals to the smooth length h is used. This means all the neighbouring particles of i must be contained in the same cell or one of the 26 ($3\times3\times3 = 27$ in total) adjacent cells (see Fig. 3.3). This reduces the time complexity of the summation from $O(N^2)$ to $O(N)$, since the average number of particles per grid cell will become a constant. In a typical SPH simulation, a

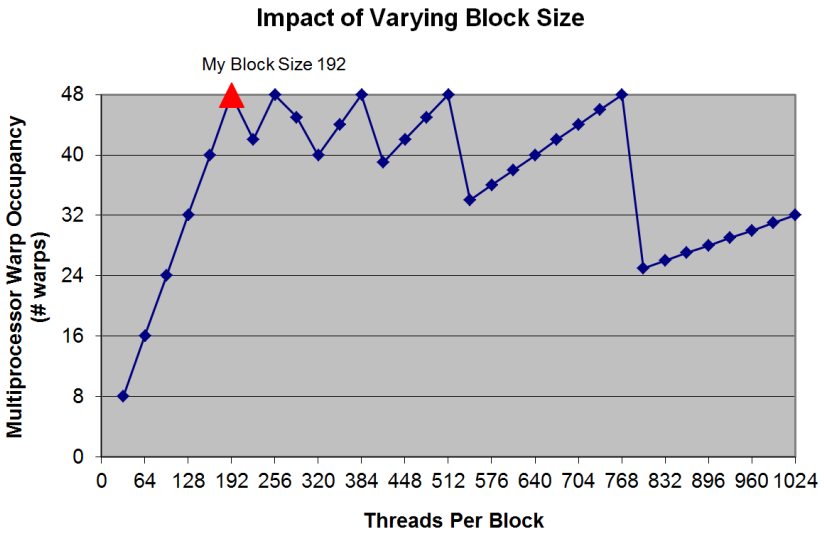


Figure 3.2: *GPU occupancy vs. threads per block.*

particle is interacting with $30 - 40$ neighbours.

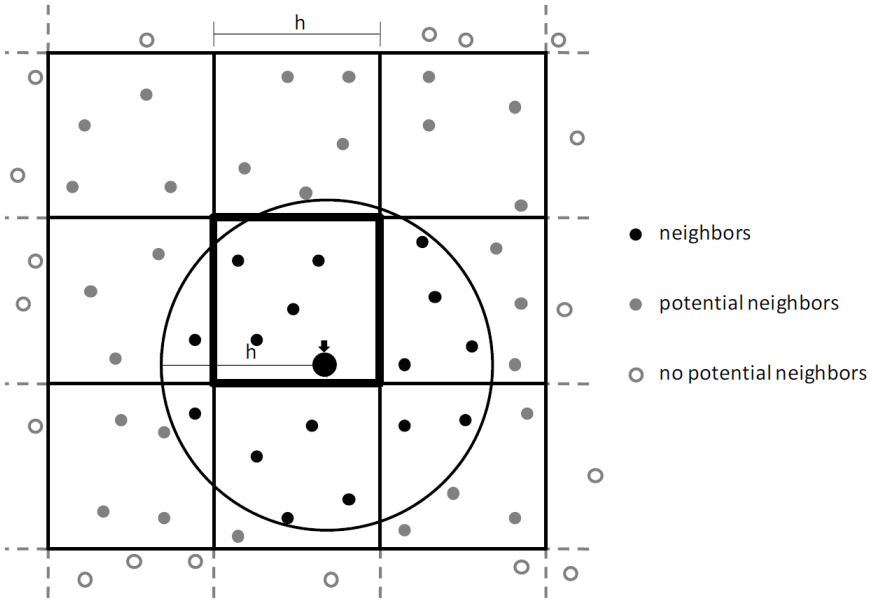


Figure 3.3: *Grid based neighbour search (Auer 2008).*

Each particle i with position $\mathbf{x} = (x, y, z)$ is inserted into a cell with coordinates $\mathbf{c} = (k, l, m)$ calculated as:

$$\mathbf{c} = ([\frac{x - x_{min}}{h}], [[\frac{y - y_{min}}{h}], [[\frac{z - z_{min}}{h}]) = (k, l, m) \qquad (3.1)$$

where $\mathbf{c}_{min} = (x_{min}, y_{min}, z_{min})$ is the minimal corner of the subdivision

space.

The cell index c is computed as:

$$c = k + l \cdot K + m \cdot K \cdot L \tag{3.2}$$

where K and L separately denote the number of cells in x and y direction. This part can be done completely in parallel since each particle computes its own index.

3.1.3 Parallel Construction

The parallel construction of the uniform grid is not straightforward since the insertion of particles into the grid might cause race conditions, i.e., two or more threads try to write to the same memory address concurrently, which is a common challenge in multi-threaded application. There are two methods to solve this problem: atomic operation or sorting.

3.1.3.1 Atomic Operation

Atomic operations are often used to prevent race conditions which allow multiple threads to update the same value in global memory simultaneously without conflicts with any other threads. Using atomic is a relatively simple algorithm for grid construction. To allow variable number of particles per grid cell, a two-pass approach is used. You can also perform only the first pass to achieve a fixed maximum number of particles per grid cell.

In the first pass, in which grid cell each particle falls is computed with Eq. 3.2 and the number of particles per grid cell is counted using an atomic increment. Two arrays are used in global memory: one stores the number of particles in each cell so far. The cell counter is initialized to 0 at each frame and updated using atomic operations corresponding to the particle positions; the other stores the particle indices for each cell, and has enough room for a large number of particles per cell. The global memory writes in this pass is essentially random (depending on

the position of the particles), and so will not be coalesced. In addition, if multiple particles write to the same cell location simultaneously, the writes will be serialized, causing performance degradation.

A parallel prefix sum operation is then performed to calculate the destination addresses for each particle. The parallel prefix sum is a standard operation in parallel computing. It is also often called a **scan** operation (Blelloch 1989). Efficient algorithms for computing the scan are described in Harris *et al.* (2007), with examples in the CUDA language.

Finally, a second pass is used to examine all the particles again, and write them to contiguous locations in the grid array using the results of the scan. This is a very similar algorithm to a single pass of a parallel radix sort.

3.1.3.2 Sorting

An alternative approach which avoids the memory conflicts is to use sorting as suggested by Kalojanov and Slusallek (2009). The particles carry several physical attributes, e.g., velocity, position and pressure. When sorting the particles, these values have to be copied several times. The memory transfer, therefore, might slow down the sorting dramatically. To avoid copying the entire particles array every time step, a secondary data structure which stores a *key-value* pair is applied, called *handle*. Each handle stores a reference to a particle (value) and its corresponding cell index (key). Sorting the handles becomes much more efficient than sorting the entire particle array since the memory usage of this structure is minimized. This *index sort* method is used in NVIDIA's CUDA-based particle system (Green 2010) and is also popular for fast parallel ray tracing of dynamic scenes (Lagae and Dutré 2008; Kalojanov and Slusallek 2009). In general, index sort is considered to be the fastest spatial acceleration methods (Ihmsen *et al.* 2011).

To represent infinite domains with low memory consumptions, a hash value can be calculated for each particle based on its cell id. There are many ways of calculating the hash value, e.g., Teschner *et al.* (2003).

However, for SPH fluids, the hash table is generally sparsely filled and a significant amount of memory is unnecessarily pre-allocated. To solve the problem, Ihmsen (2013) proposed compact hashing method which used a secondary data structure storing a compact list of non-empty (used) cells. Here the linear cell id is simply used as the hash value.

Radix sort

Green (2010) applied the fast radix sort (Satish *et al.* 2009) to perform sorting. For each byte in key, the cells are counted, scanned and the particles are shuffled accordingly. This creates a list of particle references in cell order.

Counting sort

Since the particles inside the same cell is irrelevant, there will be many duplicated keys. Therefore, it will be better to perform radix on exact cells, rather than on key bytes. Hoetzlein (2014) proposed a counting sort algorithm to achieve this. When inserting the particles into the cells, it also counts the numbers of particles in the same cell using atomic adds in parallel. By operating prefix sum on cells once, one can create the particle lists in order.

Radix sort	**Counting sort**
Assign particle to cell	Assign particle to cell
	Atomic Add for cell counts & indices
for 1 to 4 (each byte in key)	
Cell Counts	
Cell Prefix Sum	Cell Prefix Sum
Radix Add Offset And Shuffle	
Copy particles in order	Copy particles in order

Table 3.2: *Comparison of the sort algorithms.*

The comparison of the two methods is provided in Table 3.2. The counting sort method requires much less kernel calls per frame than the radix sort.

To make this sorted particle list useful, the start of any given cell in the sorted list needs to be identified. The cell index of the current

particle with the cell index of the previous particle in the sorted list. If the index is different, this indicates the start of a new cell, and the start address is written to another array using a scattered write. The end of each cell is found in a similar way.

Different from non-sorted uniform grids, a grid cell no longer stores references to all the particles in this cell. In fact, each cell just stores one reference to the first particle in the sorted array. Race conditions will not occur during this procedure. The data structure of counting sort is illustrated in Fig. 3.4.

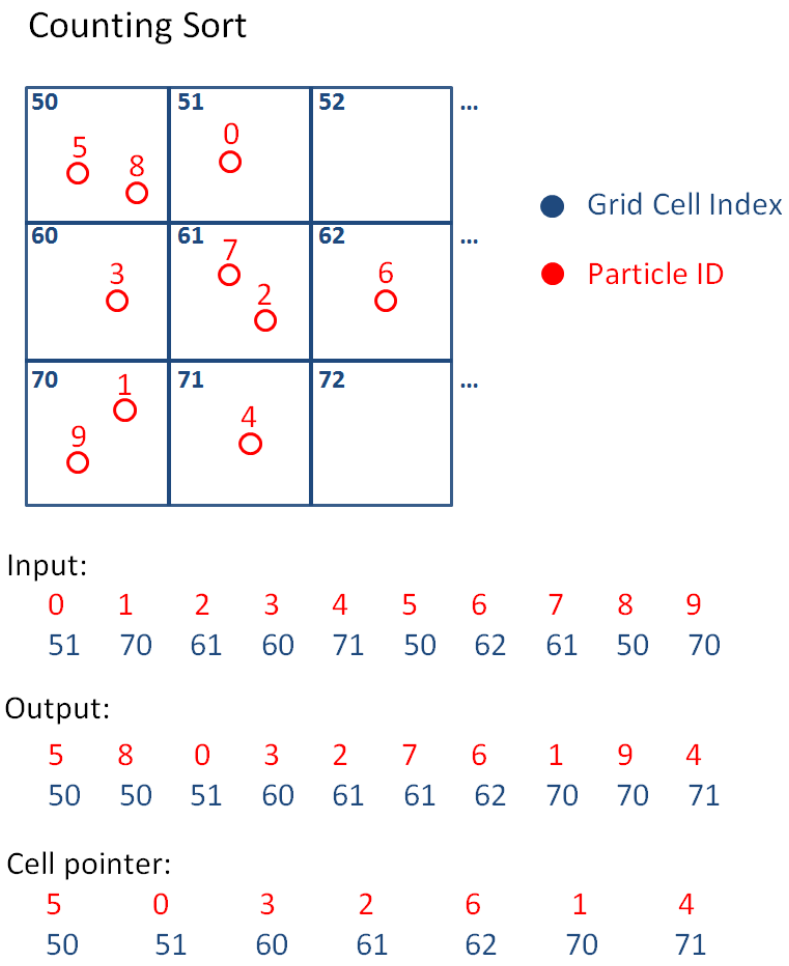


Figure 3.4: *Counting sort data structure. The diagram illustrates the uniform grid where the numbers refer to the corresponding cell indices (as in input). The output is the sorted results by cell indices. The cell pointer shows that each non-empty cell points to the first particle in the sorted particle array with corresponding cell index.*

With the sorting, particles that are in the same spatial cell are also close in memory, improving the memory coherence during queries. However, particles in adjacent cells are not necessary close in memory since the index scheme defined in Eq. 3.2 is not spatially compact (order starting from z position first). Further performance gain can be achieved by using *Z-index Sort* when computing the cell index (Ihmsen *et al.* 2014b).

3.1.4 Performance

All the results in this thesis are run on a computer with an Intel Xeon W3680 processor (8M Cache, 3.33GHz, 8GB RAM) and a GeForce GTX 560Ti GPU with 1GB on-board RAM.

The performances of both CPU and GPU methods (atomic, radix sort and counting sort) for neighbour search in SPH are tested. The simulation of 65536 particles runs at an average of 4 frames per second (fps) on CPU with uniform grid structure and 40 fps on GPU without sorting algorithm which is considerably faster. The sorting-based algorithm on GPU accelerates the method further at average 79 fps for radix sort and 138 fps for counting sort. This is because the memory access coherence is improved by sorting when performing neighbour searching, and the warp divergence is reduced — particles in the same warp tend to be close to each other in space and therefore have similar neighbours.

The memory consumption of the atomic operation method is also very sensitive to the particle distribution, while the sorting method is almost constant. The performance of atomic-based method will degrade further with random particle distribution. The counting sort method is better than the radix sort thanks to the advanced sorting process with less kernel calls per frame. The performances of the four different methods are plotted in Fig. 3.5 with the number of particles ranging from 1024 to 8 million. The counting sort method allows an interactive rate (> 20 fps) with more than 262144 particles while other methods can not. In this thesis, all the performed examples are using counting sort method.

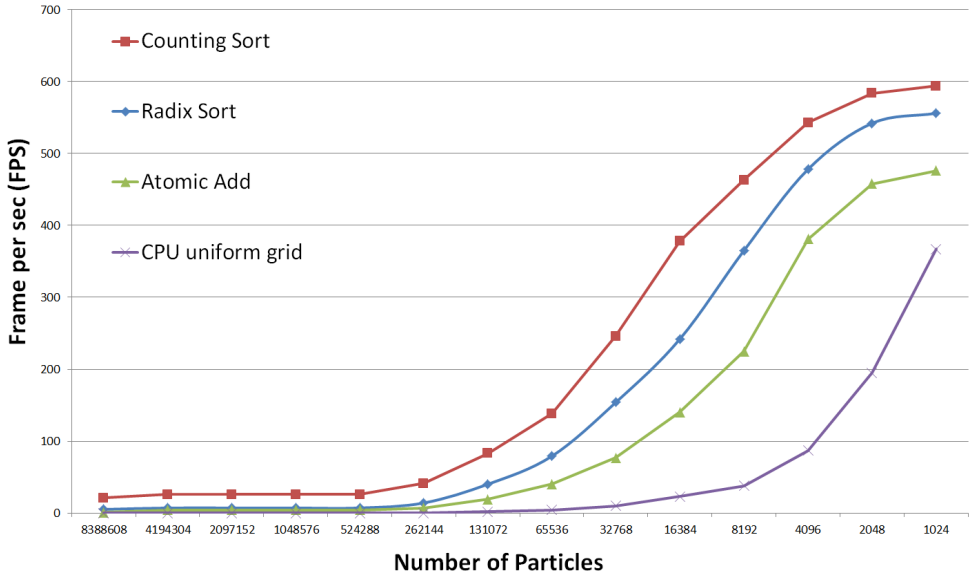


Figure 3.5: *Performances of SPH algorithms on CPU and GPU (counting sort, radix sort and atomic add). All the methods applied uniform spatial subdivision for the acceleration of neighbour search.*

3.1.5 SPH Algorithm

A basic simulation step typically calculates the density, pressure, then the various forces such as pressure, viscosity, surface tension, and gravity. The acceleration is then integrated to get the next velocity and position. The simulation process is shown in Fig. 3.6. The framework of a basic SPH algorithm is shown in Algorithm 1.

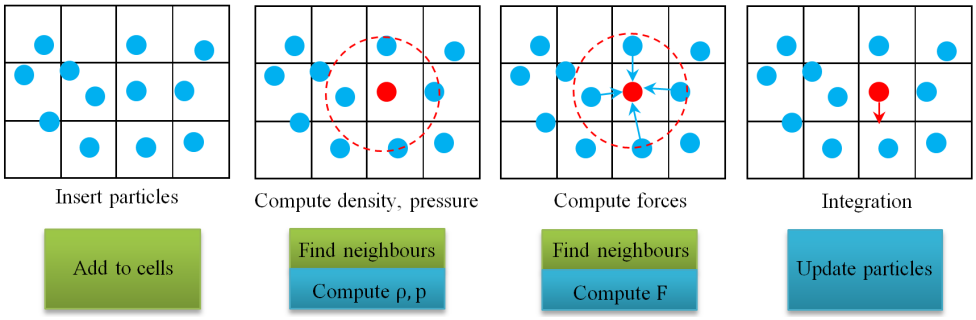


Figure 3.6: *Diagram of SPH algorithm.*

Input: Particle number N, Initial fluid particles $\mathbf{x}_{\{0,1...N\}}$, boundary level set L, count $i = 0$

Output: updated fluid particles $\mathbf{x}'_{\{0,1...N\}}$

while *time ticks* **do**
  for $i = 0$ ***to*** N **do**
    Insert particles into grid
    $\mathbf{x}_j \leftarrow$ get neighbours using uniform grids with counting sort
    $\rho \leftarrow$ calculate density using Eq. 2.19
    $p \leftarrow$ calculate pressure using Eq. 2.24
  end
  for $i = 0$ ***to*** N **do**
    $\mathbf{x}, \mathbf{v} \leftarrow$ the current position and velocity
    $\mathbf{f}^{\mathrm{p}} \leftarrow$ calculate pressure force using Eq. 2.23
    $\mathbf{f}^{\mathrm{v}} \leftarrow$ calculate viscosity force using Eq. 2.25
    $\mathbf{n} \leftarrow$ calculate normal using Eq. 2.28
    if $\mathbf{x}$ *is on the surface* **then**
      $\mathbf{f}^{\mathrm{t}} \leftarrow$ calculate surface tension force using Eq. 2.30
    else
      $\mathbf{f}^{\mathrm{t}} = 0$
    end
    $\mathbf{a} \leftarrow \mathbf{f}^{\mathrm{p}} + \mathbf{f}^{\mathrm{v}} + \mathbf{f}^{\mathrm{g}} + \mathbf{f}^{\mathrm{t}}$ using Eq. 2.7
    Integrate $\mathbf{a}$ to get new velocity $\mathbf{v}'$ and position $\mathbf{x}'$ using Eq. 2.36 and Eq. 2.37
    if $\mathbf{x}'$ *collide with* L **then**
      $\mathbf{v} \leftarrow \mathbf{v}' - \mathbf{n}(\mathbf{n} \cdot \mathbf{v}')$
    else
      $\mathbf{v} \leftarrow \mathbf{v}'$
      $\mathbf{x} \leftarrow \mathbf{x}'$
    end
  end
end

Algorithm 1: *Algorithm of SPH fluid simulation with surface tension*

3.2 Visualisation and Data Exporting

3.2.1 OpenGL Rendering

The sphere primitive look of particles in this thesis is rendered using OpenGL. Each particle can be rendered with a specific colour which provides an obvious advantage — the properties of particles such as density or pressure can be colour coded and displayed.

3.2.2 Houdini Rendering

To extract the fluid and solid surface, better rendering needs to be achieved. Houdini (SideFX 2016) is used for the final rendering of our examples. The simulation data, e.g., position, color is exported as *.geo* files which are Houdini's native geometry ASCII format. This file format can be easily read and written to, which is the reason for choosing it in this thesis. Moreover, the *.geo* file is very easy to load in Houdini and per-frame animation guarantees interactive rate (> 20 fps).

While the contents of the *.geo* file could contain all types of combinations of various attributes, primitives, groups etc., particles can be easily exported one per line as the format of

$$x\ y\ z\ w\ (r\ g\ b) \tag{3.3}$$

where x, y, z, w are the homogeneous coordinates ($w = 1$ when points and $w = 0$ when vectors) and r, g, b are the three element colors—red, green and blue. Once exported to a *.geo* file, loading in Houdini is intuitive with a $file$ sop.

Point manipulation in Houdini is a very common task and lots of enhancements could be applied to render particles in multiple forms. A geometry node called *VDB from particle fluid* is used to generate a signed distance field (SDF) representing the surface of the set of particles. Compared with the *particle fluid surface* node, it is much faster, uses less

memory and gives smoother (less lumpy) results, and has much better sheeting (narrow tendrils will appear as connected fluid, instead of a line of droplets). The VDB output can then be converted into a Polygon or PolySoup surface using the *Convert VDB* node. In brief, Houdini itself takes care of all the visualisation process; all it needs is the input simulation data.

Apart from the fluid particles, the positions of the solid particles are also exported. By using the same technique, the rigid bodies could be reconstructed in Houdini with enhancements and behave according to the imported data. During dissolution, the number of the imported solid particles is decreasing by frame, and the solute shape will change accordingly.

3.3 Summary

This chapter specifies the implementation of GPU-based SPH fluid simulation, mainly focusing on the neighbour search method for parallel construction. The GPU-based SPH implementation is applied both for the solvent and solute simulation in our dissolution model. The next chapter will start to introduce the energy-based dissolution model.

Chapter 4

Dissolution Background

As discussed in Sec. 1.1.1, dissolution simulation including the erosion is one of the most common natural phenomena which is potentially in demand for the applications in the visual effect industries. In Chapters 4, 5 and 6, a novel energy-based method is propose to approximate chemical dissolution which solves the complicated fluid–solid interaction. The method is fast, predictable and visually plausible.

4.1 Introduction

Dissolution is the result of physical and chemical processes: our dissolution model follows *Collision Theory* which explains how the chemical reaction happens. The physical and chemical reactions between fluid and solid are elegantly simulated within a single SPH solver. The Collision Theory of chemistry is used as an analogy to the dissolution process modelling. An energy function is introduced to measure the local excitation of the solute particle. Our model ensures that the dissolution result is independent of solute sampling resolution. This approach is advantageous when performing pre-visualization of the phenomena, allowing animators to quickly preview the dissolution behaviour with a low resolution sampling of the solute. A mathematical relationship between the solute features and the dissolution time is also established — allowing for

intuitive artistic control over the global dissolution rate. Our approach ensures that dissolution behaviour is physically and chemically plausible. A parallel region growing method is also proposed to deal with the separation problem during the dissolution. A large proportion of the work presented in Chapters 4, 5 and 6 was published in Jiang *et al.* (2015a).

Firstly, a literature review of the related work will be summed up, and then how the dissolution model is derived will be explained. Finally, applications of the method will be demonstrated using a number of practical examples, including antacid pills dissolving in water and hydraulic erosion of non-homogeneous terrains.

4.2 Related Work

In most observable systems, the most frequent solvent is fluid and solute is solid. There have been many studies on the physical interactions between fluids and solids. Génevaux *et al.* (2003) studied the fluid–solid interaction by representing solid with linked point masses. Batty *et al.* (2007) applied a pressure projection method with kinetic energy minimization to incorporate irregular boundary geometry into standard grid–based fluid simulations. Yet their methods are difficult to cooperate in dissolution simulation.

Carlson *et al.* (2004) and Amada (2006) treated the objects as if they were made of fluid. To maintain the object rigidity, Carlson *et al.* (2004) identified the object by velocity field, and Amada (2006) applied a corrective step to bring back the rigidity of the solid. Harada *et al.* (2007b) simulated the interaction using collision detection. To prevent fluid particles intersecting the solid boundary, both Amada (2006) and Harada *et al.* (2007b) used penalty force methods, which took several frames to push the particles out. Becker *et al.* (2009) also dealt with boundary conditions to ensure no–penetration. However, they required two additional neighbour queries for collision detection, which was expensive to compute.

Schechter and Bridson (2012) achieved the no–separation and no–slip boundary condition by sampling the air and solid surface as ghost particles. Akinci *et al.* (2012) used corrected density to compute the pressure and viscosity force to avoid the sticking artefacts. Both solved the boundary problem, making an extension to support dissolution simulation.

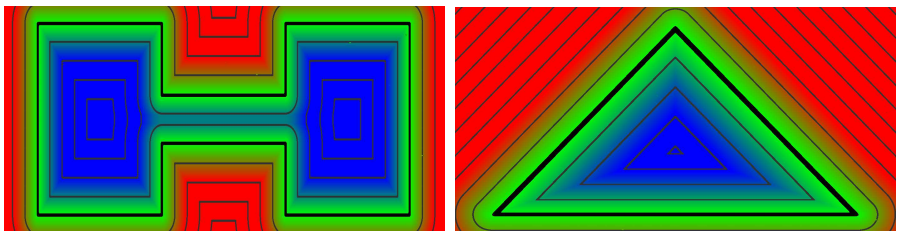
The methods above tackled the problem of physical interaction between the fluid and solid, but did not consider the chemical interaction between them. Relatively few attempts have been made to simulate the complex chemical reaction between fluid and solid.

Solenthaler *et al.* (2007) used an elasticity model for particle–based objects and modelled temperature for phase changes, allowing them to simulate objects melting in hot liquid. Stomakhin *et al.* (2014) introduced augmented MPM for phase-change and simulated butter melting in pan. Melting offers a similar visual result with dissolution, but the underlying function is a heat model in their model, which is different from the chemical model in this thesis. Our dissolution model will be extended to mimic a simplified melting effect in Sec. 11.3.

Kim *et al.* (2010) simulated bubbles around dissolving objects. Their focus was on simulating the dispersed bubble flow — the dissolution model was approximated by generating offsets of a level–set representation of the solute, which had little relationship with actual dissolution behaviour. Shin *et al.* (2010a) simulated solids dissolving in liquids by modelling solid mass transfer. They treated the solvent and solute as different fluids, and used multiple level–sets to track mixing surfaces which needed to be updated whenever the reactions and interactions occurred — an expensive but necessary step. They also did not consider about the object separation during dissolution. The latter problem is solved in Wojtan *et al.* (2007). However, they both used level–set representation to guide the object surface without additional treatment which meant their dissolution results would largely depend on the level-set grid resolution. Furthermore, all these previous approaches did not consider the relationship between the dissolution time and the mass transfer of the solute, meaning that predicting the overall dissolution time — the total

time needed for an object to be dissolved completely, was an exercise in trial and error. In this thesis, an experimental model for predicting total dissolution time from activation energy will be proposed, and the dissolution process is independent of solute sampling resolutions.

In general, methods based on level–set representations of the solute suffer from two potential problems. Firstly these methods are difficult to separate a whole object into isolated parts as a result of dissolution, implying that disconnected components will still be treated as a single body, as demonstrated in Fig. 4.1(a). Wojtan *et al.* (2007) solved this problem by representing the solid object level–set associated with a velocity field, but this required the computation of local density throughout the object. Second if the level–set has a sharp corner (C^1 discontinuity) this will be preserved in its offset surface. In the triangle example in Fig. 4.1(b) the excitation of the surrounding fluid should cause the exposed peaks to be degraded first, but the sharp angle will be retained during dissolution. Our particle–based dissolution model supports object separation and sharp feature rounding, thus achieves more physically and chemically plausible dissolution behaviour.



(a) Object separation (b) Corner rounding

Figure 4.1: *There are two significant disadvantages in using level sets to represent solutes. There is no implicit way to deal with the separation of disconnected objects (a) and sharp corners will be preserved in the offset surface (b). The blue regions show the level–set inside the object, while the red regions demonstrate the outside of the object.*

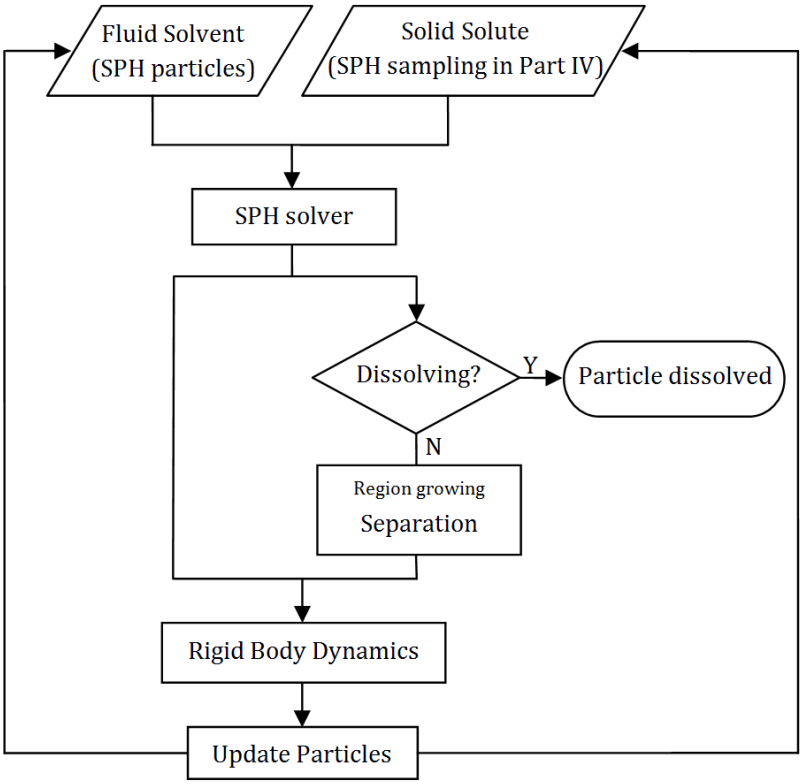
Dissolution also shares many properties with hydraulic erosion. Beneš *et al.* (2006) proposed a solution for erosion simulation by using Navier–Stokes equations, while Wojtan *et al.* (2007) animated corrosion and erosion by driving the fluids with a finite difference simulation and presenting the solid by advecting the level sets inward. Krištof *et al.* (2009)

used an SPH based method to simulate erosion, where the erosion model was adapted from an Eulerian approach. There are also researches on erosion conducting in the area of hydraulic engineering, such as Manenti *et al.* (2011) and Fourtakas and Rogers (2016). This work will demonstrate two practical erosion examples, including one which shows the specification of non–homogeneous solutes forming a natural layering stratum.

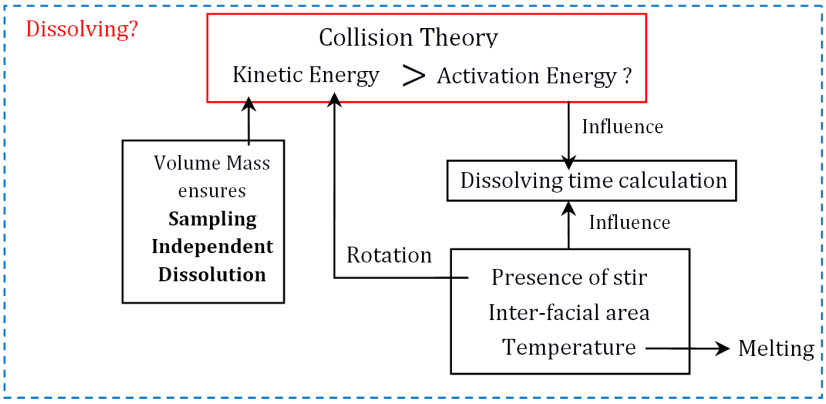
The erosion rate used in Krištof *et al.* (2009) and Wojtan *et al.* (2007) was derived from the power law, which unfortunately required a large number of parameters to control, making it impractical for use by an animator. An animator should be able to control the dissolution behaviour through a reduced set of intuitive parameters. Instead, our method requires only one single parameter which globally controls the rate of dissolution.

4.3 Summary

After identifying the problems of the previous dissolution methods and reviewing the current related work on dissolution simulation in this chapter, next chapter is going to introduce the novel energy-based dissolution model derived from chemical collision theory.



(a) Flowchart of the dissolution model.



(b) Module of decision of dissolving.

Figure 4.2: *Framework of our dissolution model.*

Chapter 5

Dissolution Model

This chapter firstly describes the *Collision Theory* and then presents our dissolution framework derived from the Collision Theory. Our approach ensures that dissolution behaviour is physically and chemically plausible. The chemical reaction between solute and solvent is derived from the Collision theory. The physical interaction between solid and fluid obeys rigid body dynamics. The dissolution model also supports features such as object separation and sharp feature rounding. The physical and chemical reactions between the solute and solvent are elegantly simulated within one single SPH solver. The calculation of the local excitation of the solute particle is derived from kinetic energy. The use of volume–corrected mass instead of mass guarantees the dissolution independent of the sampling resolution. The relationship between the dissolution configuration and the dissolution time is also experimentally deduced. The framework of our dissolution model is shown in Fig. 4.2

5.1 Chemical Collision Theory

Dissolution is a kinetic process, and in chemical engineering the dissolution of different solid substances is quantified by its dissolution rate (Kravtchenko *et al.* 1999). In Computer Graphics, to mimic the real dissolution process two different aspects of this problem needs to be con-

sidered: which part of the solute has the highest probability of being dissolved first, and the speed of dissolution — dissolution rate, represented by the mass transfer rate of the solute.

To solve these problems, it has to be understood that how chemical reactions take place and why chemical reaction rates differ (McNaught and Wilkinson 1997), which are fortunately described by **Collision Theory**, firstly proposed by Trautz (1916). Collision theory is closely related to the kinetic-molecular theory: molecules are in constant motion, occasionally colliding as they move. For a chemical reaction to occur, molecules need to collide in the right orientation and with sufficient energy (Clark 2004). The energy required for the reaction to occur is referred to as the *activation energy* — necessary to break the existing chemical bonds between substrate molecules. If the energy produced by the collision is less than the activation energy, the chemical reaction will not proceed.

Our dissolution model follows the Collision Theory maintaining the chemical plausibility of the simulation. Both the solute and the solvent are volumetrically represented with particles, with each particle becoming a representative component of the corresponding substance. That component could mean hundreds or thousands of moleculars. Therefore, collision theory of chemistry is only used as an analogy to the dissolution process modelling.

An excitation energy is introduced for solute particles which governs the dissolving process (i.e. if the particle needs to be dissolved or not). When solute particles collide with the solvent particles, energy is accumulated to the solute particle. Each solute particle carries the energy information which is summed over all the collisions which have occurred. When the excitation energy is larger than a threshold energy—the activation energy—the particle detaches from the object. This activation energy becomes an important factor in determining the dissolution rate. By detecting the relationship between the local particle excitation and the global dissolution time, it is able to control the dissolution rate during the process. The simulation continues until all the solid particles are dissolved into solvent. The solute position is updated using rigid body

dynamics while supporting object separation. This method allows us to model the dissolution process in a physically and chemically correct manner. More details will be discussed in the following chapters.

5.2 Solute and Solvent Representation

In our simulation, both the solute and the solvent are sampled with particles. Each particle is regarded as a representative component of the corresponding substance. The particle representation intuitively corresponds to the molecules in collision theory. However, one particle might represent hundreds or thousands of molecules. The particle nature also makes the tracking of object separation possible.

Solute Representation

To ensure the particles are sampled within the solute object, Signed Distance Fields (SDF) (Payne and Toga 1992) is used. SDF calculates the distance from a given point to the closest point on the object surface within a metric field (Jones *et al.* 2006) and the sign is used to indicate whether a particle is inside or outside the object. The particles inside the object will be kept and particles outside will be rejected until the desired sampling number of particles reaches. The sign is computed using Angle Weighted Pseudo Normals (Baerentzen and Aanaes 2005), which requires the closest feature. There are numerous methods to find the closest feature on a mesh from particle. Jones *et al.* (2006) provided a few methods. In this work, Axis-Aligned minimum Bounding Box (AABB) trees are used as described in CGAL (Computational Geometry Algorithms Library) for acceleration (Pierre Alliez and Wormser 2015).

Once the solute particle distribution is initialized, there is no relative movement between these particles. Therefore, the sampling of the solute can be achieved off-line before the simulation.

During dissolution, the solid particles will gradually transfer into the solution when they are activated. The dissolution always starts from boundary particles due to more frequently interactions. When it hap-

pens, the boundary particles which have high excitation energy will be dissolved, and other solid particles will naturally become the new boundary particles. All the solid particles have the possibility to become the boundary particles. To fulfil the boundary condition during the fluid-solid interaction without causing any penetration or discontinuity motion, the solid particles require an even distribution which has roughly a constant density. The evenly distributed particles can also benefit the the control of the dissolving process, guaranteeing that our dissolution process is stable and predictable. Therefore, a novel SPH sampling method is proposed for solute particle distribution which will be introduced in Chap. 9. A bunny shaped solute is sampled as Fig. 5.1. The details of the SPH sampling in dissolution simulation will be discussed in Chap. 11.

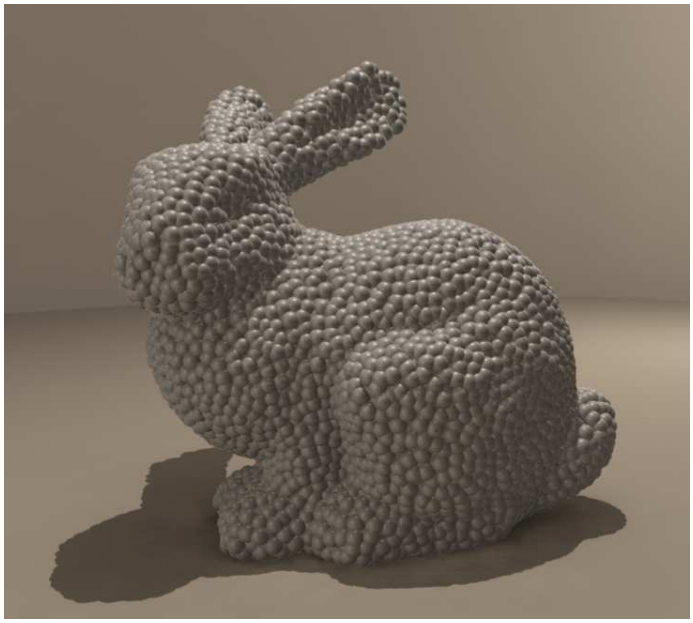


Figure 5.1: *A 3D bunny shape solute sampled using our SPH sampling method (more details in Chap. 9).*

Solvent Representation

The standard SPH is used for fluid simulation to demonstrate the dissolution framework. The implementations are discussed in Sec. 2.3 and Sec. 3.1. Other particle-based fluid methods such as alternative SPH models, PBD or MPM methods are also compatible in our framework.

To ensure the correctness of fluid-solid boundary interaction without causing any penetration or sticking artefacts, the corrected density computation of Akinci *et al.* (2012) is used to handle the boundary problem, except the entire object is sampled rather than just the surface since the boundary and shape change as the solute dissolves. By taking the same particle mass outside the summation (Hu and Adams 2006; Tartakovsky *et al.* 2007), the corrected density of a fluid particle can be written as:

$$\rho_i = m_i \sum_j W_{ij} + \sum_s \Psi_s W_{is}, \tag{5.1}$$

where j is the fluid particle in the neighbourhood of particle i, and s is the solid particle in the neighbourhood. W_{ij} and W_{is} are separately the abbreviation of $W(\mathbf{x}_i - \mathbf{x}_j, h)$ and $W(\mathbf{x}_i - \mathbf{x}_s, h)$. $\sum_s \Psi_s W_{is}$ compensates the density under-estimation for particles at the interface. Ψ_s can be calculated as:

$$\Psi_s = \rho_0 V_s = \rho_0 \frac{m_s}{\rho_s} = \rho_0 \frac{m_s}{m_s \sum_b W_{bs}} = \frac{\rho_0}{\sum_b W_{bs}} \tag{5.2}$$

where b denotes solid particle neighbours of s. Akinci *et al.* (2012) took the volume of a particle into account instead of using the mass directly. Note that even the solid particles are precomputed, a solid particle is included in the computation only if it is in the neighbourhood of a fluid particle.

Schechter and Bridson (2012) used ghost particles to handle the boundary problem. In our case, the sampled solid particle can also be seen as ghost particles. From this point of view, these two methods are actually serving one idea: the solid particles contribute to the density of the fluid particle during the fluid–solid interaction to compensate the neighbour loss of the fluid particles at the interface and achieve no-penetration results. Therefore, Schechter and Bridson (2012) is also a good choice for the boundary problem. The pressure between the fluid and solid automatically repels the fluid particles and naturally produces the mixing surface with more accuracy at lower computational cost than level–set

methods.

Each individual fluid particle carries its own attribute values such as velocity and density, which are deduced from the neighbourhood of the fluid and solid particles. These values are used to calculate the particle excitation as in Eq. 5.5 for dissolution simulation and update the physical interaction between solid and fluid (Rigid Body Dynamics) as in Sec. 5.6.

Unlike Losasso *et al.* (2006) and Shin *et al.* (2010b) required multiple level-sets methods to track numerous interfaces since they used hybrid particle and grid based methods for simulation, both solid and fluid are represented with particles in this thesis and this unified particle representation removes the need for special handling of mixing surfaces.

5.3 Sampling Independent Dissolution

Our method ensures the consistent dissolution result (within error tolerance) even when the sampling resolutions of the solute are different. This property has important applications in computer graphics: for the purposes of pre–visualisation, a solute with low sampling resolution or an adaptively sampled solute can be used to predict the dissolution behaviour of a very densely sampled solute, which improves the performance of real–time applications.

In previous methods (Wojtan *et al.* 2007; Shin *et al.* 2010a), shape alteration of a dissolving solute was achieved using mass transfer. Both methods used level–sets to represent the solute, causing the dissolution results to be largely affected by the grid resolution. The concept of **volume–corrected mass** is introduced which takes the volume and density of particle into consideration. It removes the assumption of the same mass for each particle, and calculates the mass of each particle according to its own representative volume and density. Volume–corrected mass instead of mass is used in the energy calculation for each solute particle to ensure that our result is independent with the sampling resolution. The volume–corrected mass m_{vs} of each solid particle is calculated

as:

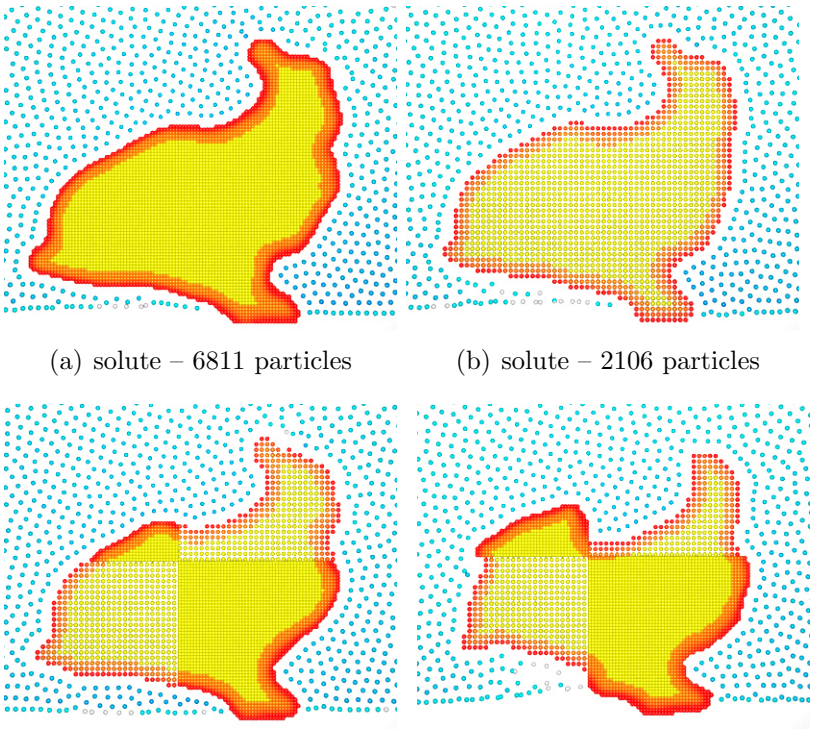
$$m_{vs} = v_s \cdot \rho_s = \frac{V}{N} \sum_b m_s W(\mathbf{x}_s - \mathbf{x}_b, h), \tag{5.3}$$

where b denotes solid particle neighbours of s. v_s and ρ_s are respectively the volume and the representative density of each solid particle. V is the volume of the entire object and m_s is the mass of each particle: both of which are constant once the object is initialised. N is the number of particles used to sample the object. It is easy to tell that $\rho_s = \sum_b m_s W(\mathbf{x}_s - \mathbf{x}_b, h)$ originates from the basic SPH density calculation in Eq. 2.19. Once the solute is sampled, the volume–corrected mass of each solute particle is set which does not change during the interactions. The volume-corrected mass of the boundary particles is set to the volume-corrected mass of the nearest interior particle to guarantee the consistency.

Fig. 5.2 demonstrates the property of sampling resolution independence by using different sampling resolutions, as well as non–homogeneous sampling resolutions, and compare these against previous dissolution methods (Wojtan *et al.* 2007; Shin *et al.* 2010a). If the same mass for each solute is directly used, each collision will result in roughly the same energy, and the solute particle at overly sampled regions will have the same possibility to be dissolved with the solute particle at sparse sampled regions, causing discontinuities like Fig. 5.2(d). Instead, with our volume–corrected mass, as in Fig. 5.2(c), the solute particles in highly sampled area have larger volume–corrected mass (due to larger density) which means higher energy with each collision, thus will be easier to be dissolved according to Eq. 5.5. This property will be further discussed in Sec. 6.2.

5.4 Particle Excitation

Particle excitation is measured by local particle energy. Since collision itself is a kinetic process, our particle excitation equation is derived from the kinetic energy $\frac{1}{2}m\mathbf{v}_r^2$, where $\mathbf{v}_r$ is the *relative velocity*. It is incorpo-



(a) solute – 6811 particles (b) solute – 2106 particles

(c) solute – 4085 particles (d) solute – 4085 particles

Figure 5.2: *Bunny shaped solutes which are sampled with different resolutions dissolve in fluid. Using our method (in (a), (b) and (c)) the sampling resolution does not affect the dissolution process, even when the sampling is non-homogeneous. (d) demonstrates the result when using previous methods (Wojtan* et al. *2007; Shin* et al. *2010a). All the images are taken from the same frame with the same initial conditions. The solute particles with high energy are coloured in red.*

rated into Eq. 2.18 to derive the expression for the particle excitation. Thanks to the use of relative velocity, the formulation yields asymmetric rules.

$$E_s = \sum_f \frac{m_f}{\rho_f}(\frac{1}{2}m_{vs}\mathbf{v}_r^2)W(\mathbf{x}_s - \mathbf{x}_f, h), \tag{5.4}$$

$$= \frac{1}{2}m_{vs}\sum_f \frac{m_f}{\rho_f}\left[||\mathbf{v}_l|| + ||\mathbf{v}_w|| + \epsilon\right]^2 W(\mathbf{x}_s - \mathbf{x}_f, h) \tag{5.5}$$

where E_s is the excitation energy of each solid particle. m_{vs} is the volume–corrected mass from Eq. 5.3; f is the solvent particle within the neighbourhood of solid particle s; m_f and ρ_f are the mass and density of fluid particle; $W(\mathbf{x}_s - \mathbf{x}_f, h)$ is a poly6 kernel (Müller *et al.* 2003), which smooths out the energy according to distance; other kernels may apply, such as cubic spline (Ihmsen *et al.* 2014b). The relative velocity $\mathbf{v}_r$ is split into three items $\mathbf{v}_l$, $\mathbf{v}_w$ and ϵ.

$\mathbf{v}_l$ is the relative linear velocity of solid and fluid particles. According to the Collision Theory, the dissolution only occurs when particles collide with right angle and enough strength. In SPH, however, particles can not collide as they will be repelled by the pressure between particles. Instead, two particles are considered to collide once one particle has entered the neighbourhood of the other. Particles outside of the neighbourhood are not computed. The size of the velocity is affected by the angle of the collision, calculated as:

$$||\mathbf{v}_l|| = \max((\mathbf{v}_f - \mathbf{v}_s)\cdot(\mathbf{x}_s - \mathbf{x}_f), 0), \tag{5.6}$$

where $\mathbf{v}_s$ and $\mathbf{v}_f$ represent the velocity of solid and fluid particles respectively. The direction of $\mathbf{v}_l$ is $\mathbf{x}_s - \mathbf{x}_f$. This is illustrated in Fig. 5.3, where only f_2 and f_3 contribute to the collision.

Fig. 5.4 shows different dissolution results with the liquid pouring from different directions. The blue particles are the fluid particles pouring from different directions. The purple particles present the dumbbell–shaped solute. The highlighted red particles show the active particles

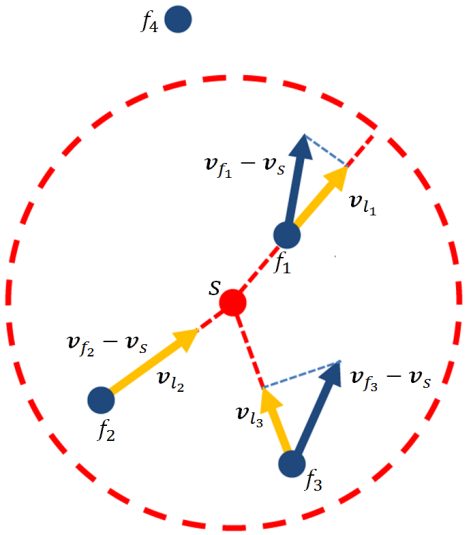


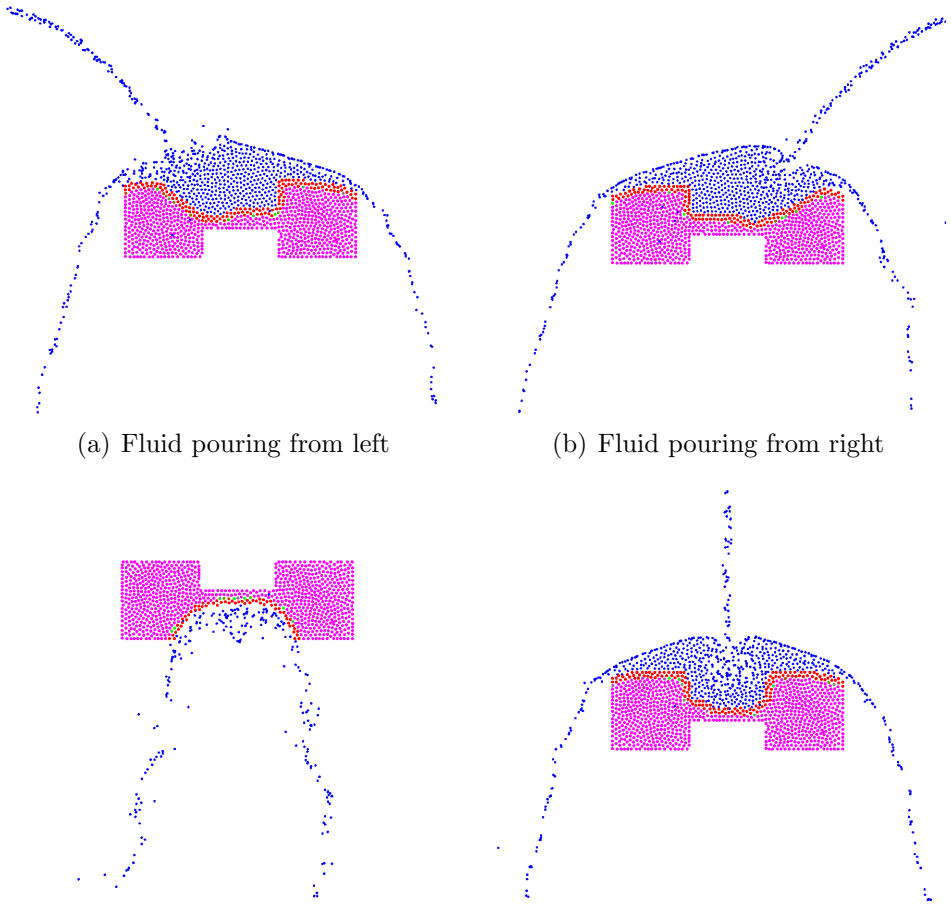
Figure 5.3: *The collision domain. s represents a solid particle and f_1, f_2, f_3, f_4 are fluid particles. The blue arrows indicate the actual relative velocity $\mathbf{v}_f - \mathbf{v}_s$, while yellow ones indicate the calculated linear relative velocity $\mathbf{v}_l$.*

which have high excitation energies.

$\mathbf{v}_w$ represents the tangential velocity due to rotation which contributes to the local particle energy. $\mathbf{v}_w$ is calculated by $\mathbf{w} \times \mathbf{r}$, where $\mathbf{w}$ is the angular velocity and $\mathbf{r}$ is the vector from the particle position to the centre of the object. This is only needed when there is obvious relative rotational motion between the object and fluid. Fig. 5.5 shows the different dissolution behaviours of the spinning bunny with and without angular velocity. It is easily noticed that the bunny with angular velocity is dissolving faster than the one without.

A small ϵ is added to represent the random thermal motion among molecules ($\epsilon = 0.00001$ in our model), which gives the possibility of dissolution when there is no macroscopic relative motion between the fluid and solid. It can be defined according to the solubility of the solute and solvent.

Therefore, the excitation energy, E_s, is the summed kinetic energy due to the neighbouring interactions and random thermal motions. It accumulates with time: when it is larger than a user specified activation energy E_0, the particle dissolves into the solvent. As described by Eq. 5.3,



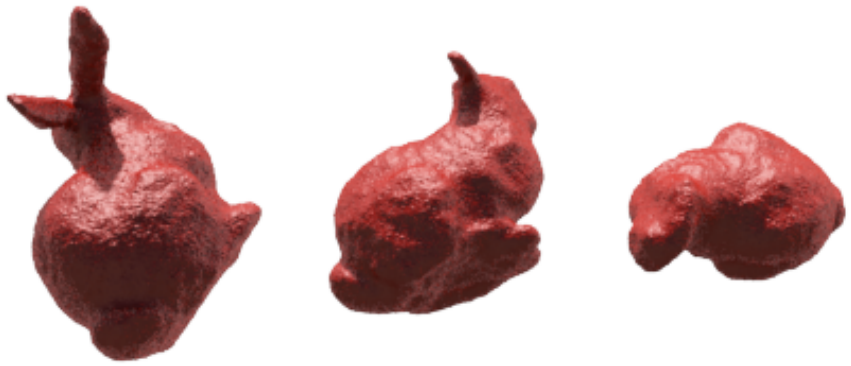
(a) Fluid pouring from left

(b) Fluid pouring from right

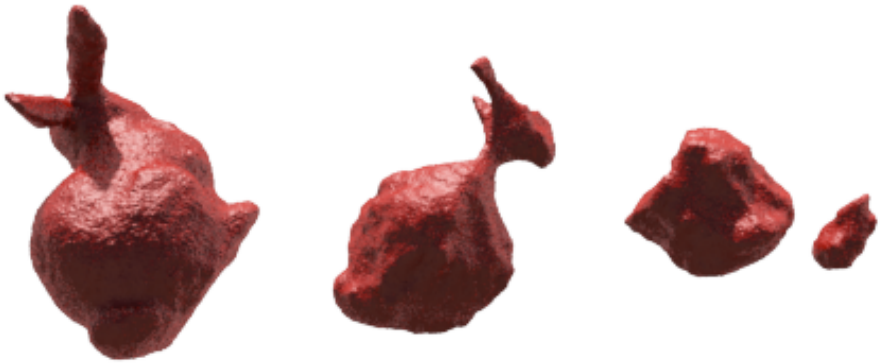
(c) Fluid pouring from bottom up

(d) Fluid pouring from top down

Figure 5.4: *Different part of the solute is dissolved due to the different solvent velocities.*



(a) Dissolving without $\mathbf{v}_w$



(b) Dissolving with $\mathbf{v}_w$

Figure 5.5: *A spinning bunny dissolving in water. Note that (b) demonstrates separation behaviour (will discuss in Sec. 5.6). The fluid is not rendered in these examples for better visualization of the solute.*

particles tend to have larger volume–corrected mass where the sampling is more compact. This corrects the single particle energy by making the particle more active (i.e. larger excitation energy) in Eq. 5.5, making it dissolve faster.

5.5 Activation Energy

The activation energy itself is derived from physical kinetic energy and the dissolution mechanism follows chemical Collision Theory. The dissolution rate of an individual particle can be controlled by the activation energy. However, this offers no control over the time taken for the object to be completely or partially dissolved: a property that is particularly

important to animators seeking to control the simulation without having to resort to trial and error.

This section provides the bridge to connect the global dissolution time with the local particle excitation. Dissolution depends on the nature of the solvent and solute, the temperature of both, the interfacial surface area between them, and the presence of mixing (McNaught and Wilkinson 1997). In our method the presence of mixing is implicitly solved by the particle excitation (e.g., Fig 5.5). For simplicity, it is assumed that temperature is a constant parameter θ (although this could be varied within more complex simulations).

The relationship between the total dissolution time T, activation energy E_0 and the interfacial surface area S is determined based on our simulation experiments. Fig. 5.6 shows the results of dissolving objects with different interfacial surface areas and different activation energies. The three objects (sphere, cube, spiky cube) have the same volume of unit 1. The surface areas are respectively 4.83, 6.0, and 8.20.

Each curve describes the dissolution times of certain object with different activation energies. The shape of the curve can be best approximated with a general exponential function ($\alpha = 0.25$ in our examples):

$$T \propto \exp(\alpha E_0), \tag{5.7}$$

By observing the dissolution time of the different objects with the same activation energy, the relationship between dissolution time T and surface area S matches:

$$T \propto \frac{1}{S}. \tag{5.8}$$

This agrees with observed behaviour: a larger surface area results in shorter dissolution time.

According to Eq. 5.7 and Eq. 5.8, it can be concluded that the relationship between the total dissolution time, activation energy and the

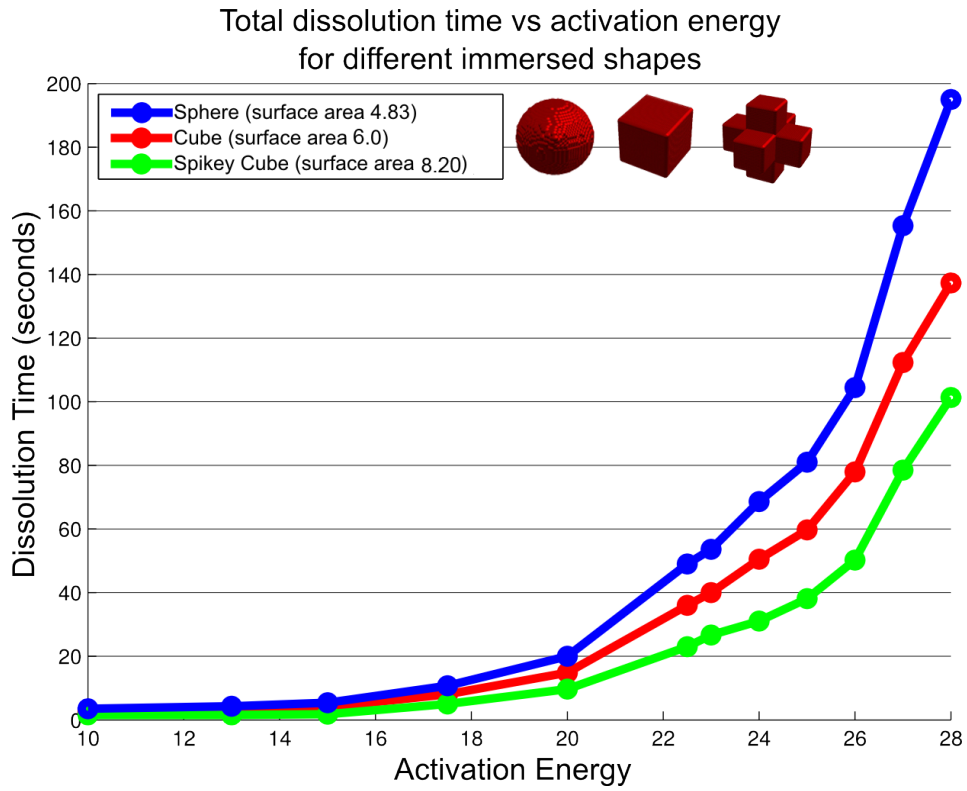


Figure 5.6: *The relationship between the total dissolution time and activation energy for different solute shapes. Shapes with more surface area in contact with the solvent naturally dissolve faster. The observed dissolution rates lead us to use the exponential function to model the total dissolution time as a function of activation energy.*

interfacial surface area can be described as:

$$T = \frac{\exp(\alpha E_0)}{\theta S}, \tag{5.9}$$

Using this experimental methodology, artists are able to better estimate the total dissolution time for objects based on the surface area and the local activation energy. An interactive tool could be developed which uses preceding simulation results of a particular fluid configuration to calculate the coefficients for better estimation. Once the coefficients are determined, the inverse of this function can be used to determine the activation energy for an artist specified target dissolution time. More complicated dissolution processes, such as a period of fast dissolution followed by a period of slow dissolution is also achievable by tuning the

activation energy at the user specified time frame.

5.6 Object Update with Separation

The positions of solutes are updated as rigid bodies. The rules of updating the solid obey the same with standard rigid body dynamics (Featherstone 2014). The transformations applied to each rigid body are divided into translational and rotational components (Harada *et al.* 2007b). The linear acceleration and angular acceleration are calculated separately.

During dissolution, parts of the object may break off and form separate shapes. Each disconnected component will be treated as a rigid body.

In order to track changes to the topology of the shape, a region growing algorithm used to detect the object separation during dissolution, a method commonly used in image segmentation (Hojjatoleslami and Kittler 1998; Baatz and Schäpe 2000). The technique is applied to identify connected regions amongst the volumetric particles. Particles that are close enough are grouped into clusters, each representing a separate rigid body.

The main challenge in implementing this algorithm is testing the proximity of particles, but as the neighbourhood information is already known as part of the fluid simulation, this becomes trivial. The neighbourhood information is updated before evaluating the region growing algorithm. The region grows from a random seed particle p by adding it to cluster C_1. Neighbours of p are added to C_1. Then all new particles in C_1 become the new seeds for the next iteration. Note that only neighbours that are not in C_1 are added. If there are no more eligible particles that can be added to C_1, a new seed is started outside the cluster C_1, add it to C_2 and expand C_2 iteratively with the neighbours, then C_3, C_4... and so on. The algorithm proceeds until all the particles are within certain cluster — each resulting cluster is a separate object. The process of the methods is shown in Fig. 5.7

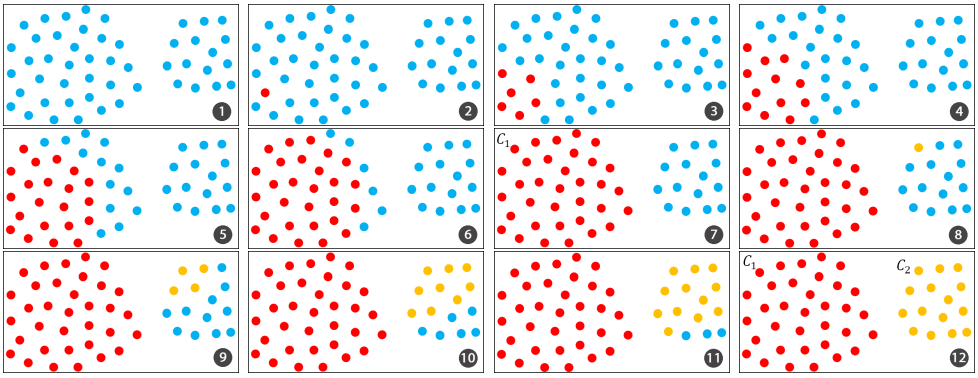


Figure 5.7: *Sequential region growing process. The particles (blue) are gradually added into C_1 (red) and C_2 (yellow). The results of the region growing method show that there are two separated objects in this example.*

Since our SPH algorithm is implemented in parallel, the sequential region growing method may become the bottleneck of our framework with time complexity $O(N)$, a new parallel region growing algorithm on the GPU is proposed which suits well with our dissolution framework. Our GPU accelerated region growing algorithm is derived from Happ *et al.* (2013) and improves the time complexity to an average of $O(log(N))$.

Initially, all the particles are considered as seeds, one seed representing one class, shown in Fig. 5.8. Each parallel thread assigned to individual particle merges adjacent particles iteratively to minimize the class sizes. To avoid race conditions described in Sec. 3.1.3, every time-step the particle's class is updated as the lowest class of the neighbouring particles (including itself) according to the statement of the previous frame. The algorithm stops when there is no change of the classes.

The GPU region growing method avoids the data transfer in-between CPU and GPU during the separation check every frame. Compared with CPU region growing, the performance on the GPU improves with the number of the separation parts. An extreme situation is that each particle represents an isolated object, in which case the time complexity reaches $O(1)$. The performance is also affected by the size of each separated part. In general, the smaller the number of particles, the faster the algorithm. The overall computational cost is constrained by the largest separated part. The performances of the CPU and GPU region grow-

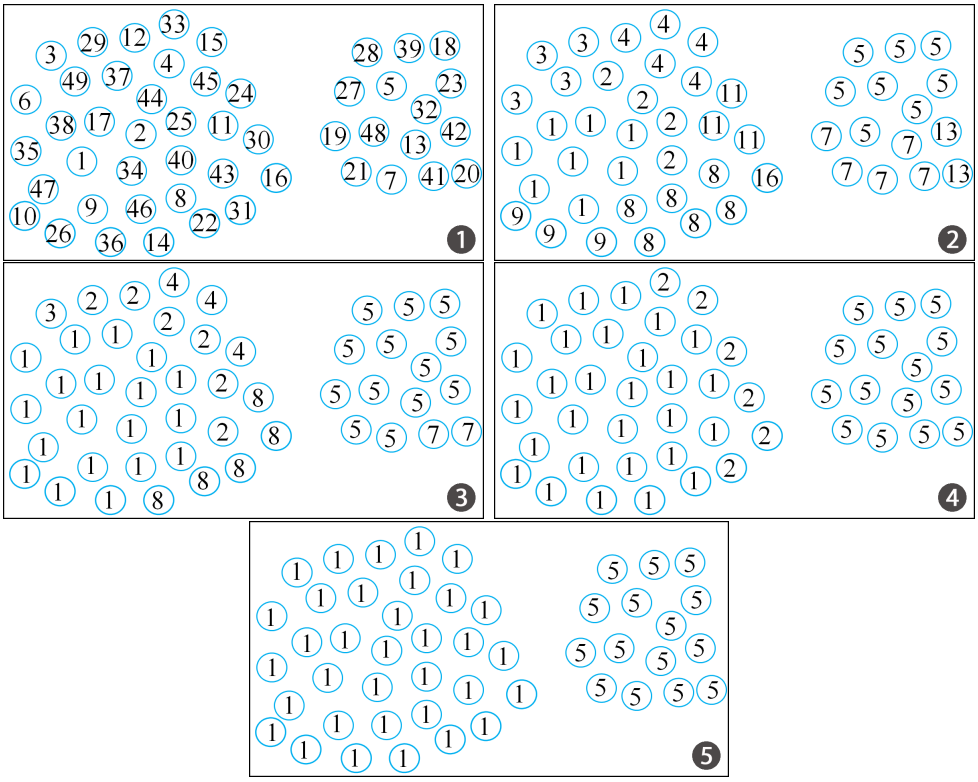


Figure 5.8: *Parallel region growing process. Each particle is initially labelled with one class. Afterwards, they gradually merge class with their neighbours as the lower class. The results show that there are two isolated objects with label* 1 *and* 5.

ing methods are compared using a simple test, in which there are two objects, and each with 70000 particles. Given the same neighbouring information, the CPU region growing requires $31ms$ to converge while GPU region growing takes $15ms$ which is more than 2 times faster. With four separations and each with 35000 particle, GPU algorithm maintains the speed with $16ms$ while CPU goes up to $53ms$. Further discussion and performance improvement of the algorithm is an interesting area of future research.

5.7 Summary

In this chapter, a novel dissolution model is introduced. The key to this model is a new expression for particle excitation. By using the volume–corrected mass instead of mass to update local particle energy, it is able to ensure that the dissolution results are consistent and independent of the sampling density. A mathematical relationship between the activation energy, the interfacial surface area, and the total dissolution time is also established, allowing animators to easily adjust the global properties of the dissolution simulation by modifying a single parameter. Our model also supports object separation during dissolution by using a parallel region growing method. The applications and results of the dissolution model will be demonstrated in the next chapter.

Chapter 6

Applications and Results of Dissolution Simulations

The simulations are implemented in CUDA as described in Sec. 3.1. Real–time results are rendered in OpenGL and simulation results are rendered using *Houdini* as discussed in Sec. 3.2. The computational cost of our dissolution model depends on the number of both solute and solvent particles, the complexity of interaction between solute and solvent and the update of solute positions such as separation and rotation. The performance of our technique in different examples is presented in Table 6.1. In the same example of *Spheres*, the performance is largely improved with less solute particles, which proves that our sampling independent dissolution model allows animators to preview the dissolution behaviour with a much lower computational cost and is therefore useful for pre-visualization.

6.1 Solute Dissolves in Fluid

Antacid pills in water is a classic demonstration of dissolution behaviour for simulation. Fig. 6.1 demonstrates the dissolution effect of two antacid pills dropped into a glass of water. Bubbles caused by the release of trapped air and chemical processes are recognisable clues that dissolution

Example	*Figure*	*Solute*	*Solvent*	*FPS*
Spinning bunny	Fig. 5.5	28553	65535	27
Antacid pills	Fig. 6.1	12745	80000	50
Bunny with bubbles	Fig. 6.2	11000	80000	52
Spheres (low)	Fig. 6.3(a)	8538	100000	148
Spheres (high)	Fig. 6.3(b)	22274	100000	48
SCA Logo	Fig. 6.4	1000	3000	355
River Bed	Fig. 6.5	15000	385948	19
Layered Terrain	Fig. 6.6	169650	262140	20

Table 6.1: *The performance in average frames per second (FPS) of our dissolution model in different examples.* Solute *and* Solvent *represent the number of particles in each.*

is taking place. Bubbles are added at the dissolution site to increase the authenticity of the simulation. The bubble motion is modelled using basic Brownian motion (Kim *et al.* 2010). The direction of the bubble's velocity is altered while the bubble particle is determined to be scattered. The new direction of the bubble particle can be computed as:

$$\cos\vartheta = \frac{2\xi + \beta - 1}{2k\xi - \beta + 1}, \tag{6.1}$$

where ϑ is the altered direction in radians, $\xi \in [0, 1]$ is a uniform random number and $\beta \in [-1, 1]$ is the scattering coefficient ($\beta = 0.1$ for these results).

Fig. 6.2 shows a bunny dissolving in fluid with the same bubble simulation as the antacid pills.

6.2 Sampling Independent Dissolution

To further prove the sampling independence property of our dissolution model, the same solute with two different sampling resolutions is tested under the same situations. In Fig. 6.3, liquid is pouring over a sphere, demonstrating the effect of pouring fluid at a targeted area. One of the spheres is sampled with 8538 particles while the other with 22274 particles. The percentage of the volume difference is calculated at each frame of these two spheres: with our method, the difference reaches a

Figure 6.1: *Two antacid pills dissolving in water. The bubbles are generated at dissolution sites and animated using basic Brownian motion. The buoyancy of the objects is not considered in the examples.*

Figure 6.2: *A bunny dissolving in fluid.*

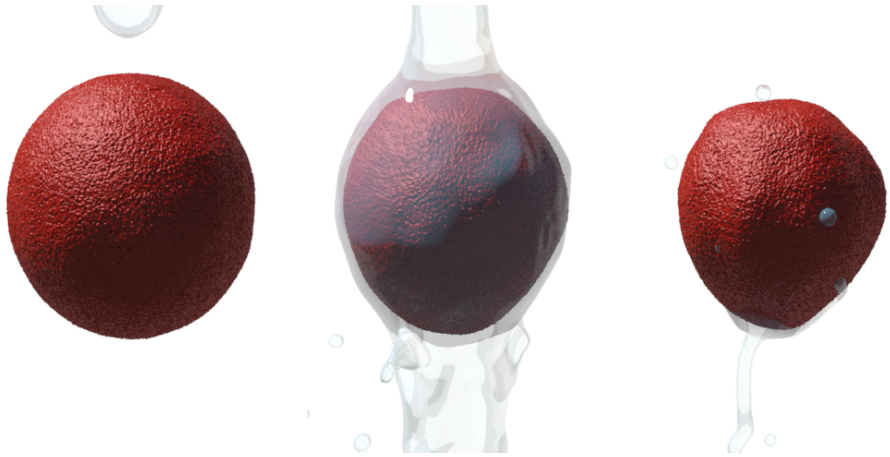
maximum of 3.4% of the entire sphere volume, while with the method of Wojtan *et al.* (2007) and Shin *et al.* (2010a) the low sampled sphere dissolves completely before the high sampled sphere halves in size. This example demonstrates that our dissolution results are consistent with different sampling resolutions.

To ensure that the fluid realistically adheres to the sides of the sphere, the surface tension model of Akinci *et al.* (2013) is used.

6.3 Separation during Dissolution

In Fig. 5.5(b) the bunny separates into pieces as a result of the dissolution. Fig. 6.4 shows the image sequence of object separation during dissolution. The SCA logo dissolves in liquid. Three letters gradually

(a) Sphere sampled with 8538 particles



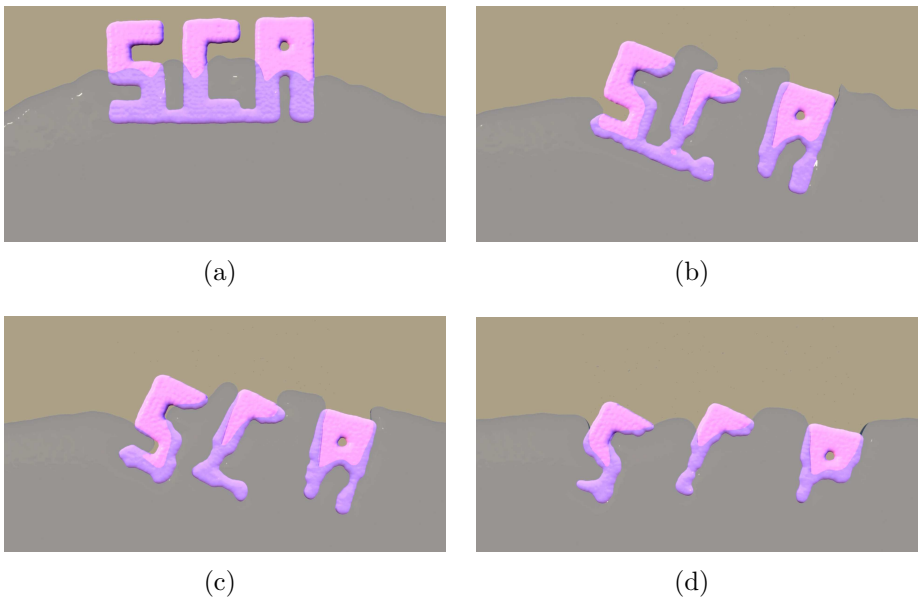
(b) Sphere sampled with 22274 particles

Figure 6.3: *Fluid is poured over spheres with different sampling resolutions, with almost identical results, except that the solute appears smoother with a higher sampling resolution.*

detach from each other and move freely after separated.

6.4 Erosion

Our dissolution method can also be applied to simulate hydraulic erosion phenomena. While the actual physical properties of different terrain types are not used, these examples demonstrate the viability of using our model for practical simulations. Fig. 6.5 shows the hydraulic erosion acting on the floor of a river bed. As the fluid runs over the terrain, certain



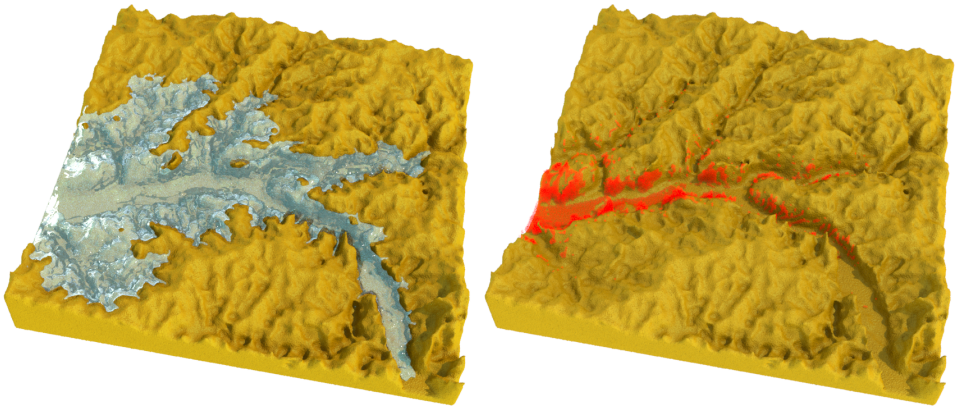
(a) (b) (c) (d)

Figure 6.4: *Dissolving SCA Logo. In (a) SCA logo drops into a liquid. As the object dissolves, the letters become detached ((b) and (c)), after which disconnected components move freely and dissolve gradually (d).*

soil particles become dissolved in the water. Note that more complex properties of erosion, such as fluid carrying and relocating sediment, are ignored in this simulation.

One advantage of our energy based model is that different terrain types can be simulated by simply modifying the activation energy of individual layers of particles. This allows us to simulate erosion of rocks consisting of different layers each with different material properties, giving rise to natural formations similar to those which can be found in mountainous or coastal regions.

This simulation is demonstrated in Fig. 6.6, where a three layered terrain is immersed in turbulent fluid to simulate coastal erosion. Different activation energies are assigned to particles in each layer, with the lowest activation energy assigned to the middle layer — leading to a natural arch formation.



(a) Fluid simulation (b) Resulting erosion

Figure 6.5: *Hydraulic erosion using our dissolution model. (a) shows the dissolution process, while in (b) the removed soil particles are shown in red.*

6.5 Summary

This chapter demonstrates that our dissolution model achieves visually plausible dissolution animations and erosion behaviours by way of examples.

In Chapters 4, 5 and 6, the novel energy-based dissolution model derived from the Collision Theory has been introduced. This method is straightforward to implement in an existing particle–based fluid framework, and naturally lends itself to a parallel implementation as discussed in Chap. 3.

The dissolution simulation depends on the solvent simulation as well as the solute simulation. Although our dissolution model is independent of sampling resolution of the solute, the sampling quality of the solute particles does affect the visual plausibility and simulation correctness. A new blue noise sampling method — SPH sampling will be proposed in the next chapter for the particle distribution of the solutes.

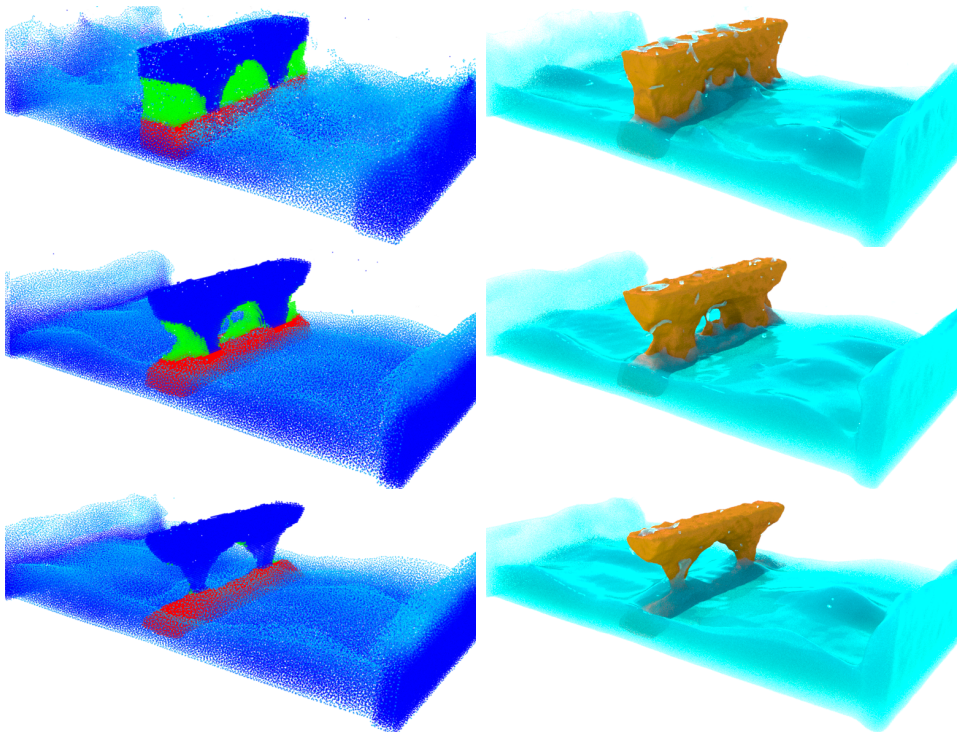


Figure 6.6: *A simulation of layered terrain erosion using our dissolution model. The solute consists of three different material layers, each with a different activation energy. The green layer has the lowest activation energy which leading to a faster dissolving. Forces are added to the fluid to keep the fluid active. The rendered results are shown in the right corresponding to the particle representations.*

Chapter 7

Blue Noise Sampling Background

7.1 Introduction

Although Smoothed Particle Hydrodynamics (SPH) was initially introduced to solve problems in astrophysics, it is now widely used in fluid simulation. As discussed in Chap. 2, SPH is not a specific solver, but an interpolation method for particle systems. It is found that this results in sample points (i.e. particles) with a high quality blue–noise spectrum. Inspired by this a method for sampling derived from SPH is developed. In the following chapters, a novel algorithm for blue noise sampling using a tailored SPH method is introduced so that specific regions possess blue noise characteristics.

Fast sampling is achieved in general dimensions for curves, surfaces and volumes. By varying a single parameter our method can generate a variety of controllable blue noise sample patterns. Our method supports adaptive sampling and multi–class sampling, and is suitable for applications such as image stippling and remeshing. A large proportion of the work presented in Chapters 8, 9 and 10 was published in Jiang *et al.* (2015b).

The following chapters are structured as follows: Chap. 7 reviews ex-

isting sampling methods and the motivation of using SPH in sampling. The connections between SPH interpolation and traditional sampling methods are explained in Sec. 8.1, where the convergence criterion for SPH sampling is also stated. Sec. 8.2 analyses the trade-off between noise and aliasing, and performs experimental studies of the SPH kernel and its influence on the distribution properties of samples. SPH sampling schemes and the differences with the traditional SPH fluid simulation method are subsequently discussed in Chap. 9. The applications of SPH sampling are given in Chap. 10, while the use of SPH sampling in dissolution framework for the solute particle distribution is finally discussed in Chap. 11.

7.2 Colour of Noises

The nature of various types of noise patterns is usually described in terms of the Fourier spectrum color (Zhou *et al.* 2012). The colors of noise are often (but not always) given in reference to the color of light with similar spectrum. Different colors of noise have different spectral distribution properties. The most well known example is **white noise**, who has a flat power spectrum, with equal power distribution within any equal interval of frequencies. The samples of white noise are independently random variables with uniform probability over certain interval. The samples in our examples are initialized as white noise. **Blue noise** generates samples that are stochastic yet evenly dispersed in the spatial domain. Fourier spectrum analysis shows that the spectral energy of such sample points is largely absent in the low-frequency region (i.e. 'red' in terms of the visible spectrum), while evenly spread in the high-frequency region (i.e. blue). This 'blue noise' property turns out to greatly benefit anti-aliasing, and the sample patterns are also visually pleasing, leading to its popularity in the aforementioned applications, such as image synthesis (Li *et al.* 2010b), non-photorealistic rendering (Sun *et al.* 2013), and geometry processing (Bowers *et al.* 2010). **Pink noise** is a signal with a frequency spectrum such that the power spectral density is inversely proportional to the frequency of the signal. Its spec-

tral energy is concentrated in the low-frequency bands. Samples of pink noise visually form concentrated clusters. It can be used to characterize the distribution of plants (Condit *et al.* 2000) or galaxies (Martinez *et al.* 1995) and is also present in the statistics of DNA sequences in biological systems (Josephson 1995). **Green noise** consists of primarily mid-range frequencies. It characterizes the distributions of natural settings, without man-made noise (Schroeder 2012), such as plumed seeds (Greene and Johnson 1989). It can be used for half-toning (Ulichney 1988; Lau *et al.* 1998; Pang *et al.* 2008), benefiting printing devices which prefer clustered-dot rather than single-dot patterns. In this chapter a method to generate a variety of blue noise profiles is developed.

7.3 Blue Noise Sampling

One of the fundamental problems in computer graphics is aliasing which is often unavoidable since most of an image signal is not band-limited. One common solution to reduce the visibility of this aliasing is to employ sampling patterns with a blue noise power spectrum, which prevents conspicuous aliasing artefacts by replacing them with incoherent noise.

There are a variety of methods available for generating blue noise samples. One of the earliest blue noise sampling algorithm is the *dart throwing* method, which generates random samples with minimum distance constraints (Dippé and Wold 1985; Cook 1986; Mitchell 1987), also known as Poisson disk distributions — only points whose surrounding disk do not overlap with any other disk laid so far are accepted (McCool and Fiume 1992). The original algorithm is computationally expensive since the rejection count could be large, and it is difficult to precisely control the number of samples. Since then, various alternatives have been proposed to improve the performance of dart throwing (Dunbar and Humphreys 2006; Wei 2008; Öztireli *et al.* 2010; Kalantari and Sen 2011; Guo *et al.* 2015).

Relaxation–based methods introduce additional topological information into the point pattern, making it easier to precisely control the

number of samples and improve the spectrum properties. A well–known approach in this class is the Lloyd's method (Lloyd 1982), also known as Voronoi relaxation, which was firstly introduced to computer graphics by McCool and Fiume (1992). Despite its popularity, Lloyd's method suffers from two main drawbacks. McCool and Fiume (1992) already found that sampling produced by Voronoi relaxation has too much regularity which causes aliasing problem for intended applications. The other shortcoming, as indicated later by Balzer *et al.* (2009), is the imprecision of approximating the density functions, where an obvious blurred density occurs.

Since then, several techniques have emerged to overcome the problem, include the *tiling-based* method, which allows progressive refinement and accelerates the speed to real-time by pre-computing tile patches. This is initially proposed by Shade *et al.* (2000), they pre-constructed these tiling blocks such that they obey the minimal distance requirement with the points of all possible neighbouring tiles. This idea is later refined with spatially adapted distributions, such as Ostromoukhov *et al.* (2004); Kopf *et al.* (2006); Lagae and Dutré (2006); Ostromoukhov (2007); Wachtel *et al.* (2014). In addition to these complex construction methods, recent techniques show significant progress in the generation of blue noises based on the idea of Lloyd's method.

Balzer *et al.* (2009) presented a variant of Lloyd's method by imposing capacity constraints over the Voronoi tessellation — Capacity Constrained Voronoi Tessellations (CCVT), which solved the regularity issue. However, CCVT is considerably more expensive to compute than Lloyd's method, as it needs to preserve the capacity constraints at every iterative step. An improved implementation of CCVT, called Fast CCVT (Li *et al.* 2010a), increased the speed while still maintaining the basic discrete sampling technique. More recently, Xu *et al.* (2011) proposed Capacity-Constrained Delaunay Triangulation (CCDT), replacing the Voronoi cells of CCVT with Delaunay triangles on the same sites to avoid the dual problem. de Goes *et al.* (2012) modelled CCVT as a constrained transport problem. Similarly Capacity–Constrained Surface Triangulation (CCST) (Xu *et al.* 2012) extended CCDT to allow for

surface sampling with surface area constraints. Compared with CCVT-like methods which focus on one kind of sampling pattern, our method provides controllable sampling pattern and can provide better trade-off between effective Nyquist Frequency and oscillation with a much faster speed.

Using a kernel density model, Fattal (2011) proposed a new sampling algorithm with excellent blue noise characteristic and linear time complexity. His method introduced a 'randomness' parameter to avoid hexagonal patterns. However, it is not capable of varying the samples' distribution properties. Our method, in contrast, allows users to choose the desired blue noise patterns with the change of one parameter. Although both use kernel functions to define the density, the density in SPH implicitly considers the volume, and the use of pressure term ensures the volume preservation. This is analogous to the capacity constraint in CCVT, and thus our method naturally avoids hexagonal patterns without using the 'randomness' parameter. In addition, Fattal's method changes the number of samples dynamically to match the point density with the target density, resulting in difficulties of controlling the exact number of samples. Our method, on the other hand, allows user-specified number of samples.

7.4 Sampling with Controllable Spectrum

The ability to control the spectral profile is important since different applications may perform better with alternative sample patterns.

In Zhou *et al.* (2012) and Öztireli and Gross (2012), gradient-based methods are presented to generate various controllable blue noises. Wachtel *et al.* (2014) proposed a tile-based method to achieve the same capability. To improve stochastic integration, other research has proposed blue noise with controlled bias, variance (Subr and Kautz 2013) and aliasing (Heck *et al.* 2013). These techniques are flexible and can generate samples with a range of different distribution properties.

Others have proposed methods to analyse and measure the quality and distribution properties of sampling points in various domains (spatial and spectrum analysis) (Dippé and Wold 1985; Shirley 1991; Dale *et al.* 2002; Wei and Wang 2011), and their anti-aliasing properties (Heck *et al.* 2013; Subr and Kautz 2013). These provides scalar measures to indicate the global uniformity.

In terms of spectrum analysis, one of the most powerful and widely adopted tools is the Fourier power spectrum (Ulichney 1987; Lagae and Dutré 2008). Two statistics derived from the power spectrum, radial means and anisotropy graphs, are the two important statistics to quantify and detect artefacts in uniformly distributed samples, which are used for evaluations in this thesis.

The Fourier transform (Bracewell 1965) of a set of N samples $\{s_k\}(k \in [0, N-1])$ is defined as:

$$F(\mathbf{w}) = \sum_{k=0}^{N-1} e^{-i2\pi \mathbf{w} s_k} = \sum_{k=0}^{N-1} cos2\pi \mathbf{w} s_k + i \sum_{k=0}^{N-1} sin2\pi \mathbf{w} s_k \tag{7.1}$$

where $\mathbf{w}$ is the frequency vector (in 2D, $\mathbf{w} = [w_x, w_y]$ which represents the scalar frequencies in the x and y directions respectively). For the purpose of frequency analysing, the Fourier power spectrum is the main focus. It is further divided by N in order to normalize it. The power spectrum is calculated as:

$$P(\mathbf{w}) = \frac{|F(\mathbf{w})|^2}{N} = \frac{1}{N}\left[\left(\sum_{k=0}^{N-1} cos2\pi \mathbf{w} s_k\right)^2 + \left(\sum_{k=0}^{N-1} sin2\pi \mathbf{w} s_k\right)^2\right] \tag{7.2}$$

The average and relative variance of $P(\mathbf{w})$ in each frequency band (consisting of frequency vectors at the same magnitude but different orientations) provides the radial means and anisotropy measure. They are typically plotted into 1D graphs and used to analyse the samples' spectral distribution properties. Fig. 7.1 shows a typical power spectrum of a blue noise distribution and the corresponding radial mean and anisotropy.

The radial mean R_m is defined as the average of $P(\mathbf{w})$ within a given frequency range. When the range is sufficiently small, it approximates the expected value of $P(\mathbf{w})$ at a specific frequency $\mathbf{w}$. In other words, $R_m = E[P(\mathbf{w})]$, where E denotes the expected value with respect to the joint distribution of the samples. R_m measures the spatial artefacts of the distribution which is further quantified by its effective Nyquist frequency and oscillation.

The anisotropy is defined as the variance of $P(\mathbf{w})$, which can be calculated as $E[P^2(\mathbf{w})] - E^2[P(\mathbf{w})]$. It measures the angular biases of the distribution. A good quality of blue noise sampling should have an almost flat anisotropy which means there is almost no difference in any orientation of the frequency vectors. An idealization of blue noise is called **step blue noise** (Heck *et al.* 2013). The power spectrum of step blue noise is zero in low frequencies and constant in high frequencies, shown in Fig. 7.2.

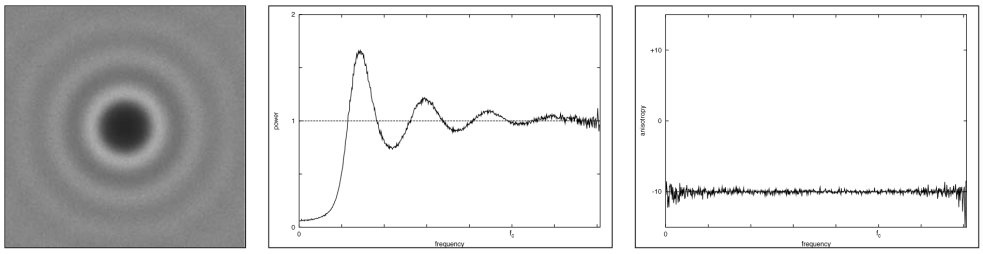


Figure 7.1: *A typical power spectrum (left), radial mean (middle) and anisotropy of blue noise sampling (right). (Reproduced from (Lagae and Dutré 2008))*

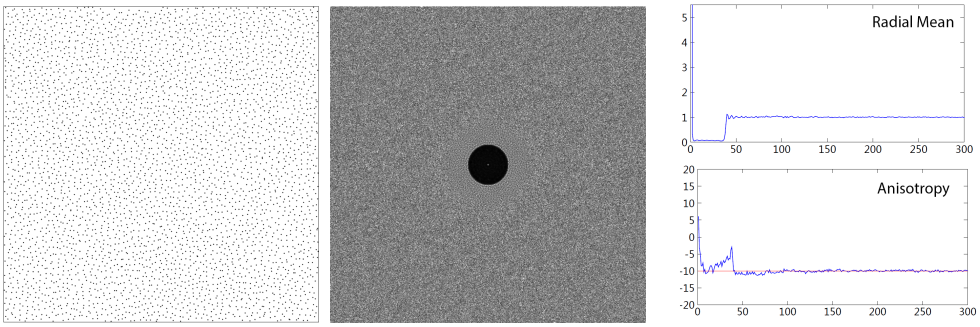


Figure 7.2: *Ideal blue noise profile with its point distribution, power spectrum, radial mean and anisotropy. Reproduced from Zhou* et al. *(2012).*

To numerically assess the blue noise qualities from the radial mean of the Fourier spectrum, the measures in Heck *et al.* (2013) are applied which characterized the blue noise sampling property by its effective Nyquist frequency ν_{eff} and oscillation Ω. The effective Nyquist frequency measures the size of the empty low frequency region in its Fourier power spectrum. A blue noise sampling with high ν_{eff} can effectively reduce the low frequency noise. On the other hand, oscillation Ω measures the variation in the non-zero region of a blue noise spectrum. High oscillation would result in structured aliasing. Ideally the perfect blue noise shall have high ν_{eff} with low Ω (Heck *et al.* 2013). The effective Nyquist frequency and oscillation on the radial mean curve is shown in Fig. 7.3.

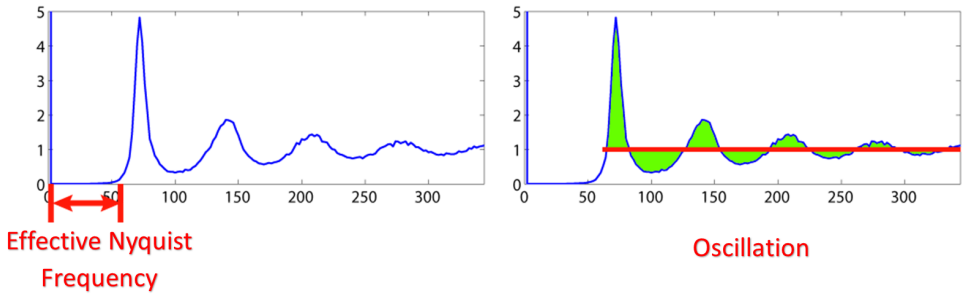


Figure 7.3: *The effective Nyquist frequency corresponds to the empty region in the low frequency part. Oscillation refers to the oscillation after the initial jump, which is marked as green.*

Recently, Wei and Wang (2011) observed that Fourier spectrum in Eq. 7.2 can be reformulated into a new form as Eq. 7.3. $p(\mathbf{d})$ in Eq. 7.4 is called the *Differential Distribution Function* (DDF) in differential domain Ω_d.

$$P(\mathbf{w}) = \frac{1}{N}\sum_{k=0}^{N-1}\sum_{j=0}^{N-1} cos\left[2\pi\mathbf{w}\cdot(s_k - s_j)\right] \tag{7.3}$$

$$\approx \int_{\Omega_d} cos(2\pi\mathbf{w}\cdot\mathbf{d})p(\mathbf{d})\delta\mathbf{d} \tag{7.4}$$

DDF describes the probability distribution of the difference $s_k - s_j$

between every pair of samples k and j. It can be calculated as:

$$p(\mathbf{d}) = \frac{1}{N}\sum_{k=0}^{N-1}\sum_{j=0}^{N-1}\kappa(\mathbf{d},(s_k - s_j)) \tag{7.5}$$

$$= \kappa(\mathbf{d},0) + \frac{1}{N}\sum_{k=0}^{N-1}\sum_{\substack{j=0\\ j\neq k}}^{N-1}\kappa(\mathbf{d},(s_k - s_j)) \tag{7.6}$$

where κ is a density estimation kernel similar with the kernel discussed in Sec. 2.3.5. Without the need of a Fourier transform, one can calculate the spectral properties through the samples' local spatial statistic. Thanks to its local character, DDF accounts for spatial distance metric changes, hence allows non-uniform sample analysis. In this thesis, DDF is used for the spectrum analysis of the adaptive sampling.

7.5 SPH in Sampling

SPH is a well-known method in fluid simulation — it computes particle distributions to minimize the internal pressure variance. Experiments will show that this results in sample points with a high–quality blue noise spectrum, show in Fig. 8.3. To the best of our knowledge, this is first attempt to apply SPH to the problem of sampling. Previously, there are few fluid papers associated SPH with sampling idea. Adams *et al.* (2007) proposed adaptive SPH. de Goes *et al.* (2012) related SPH model to a power diagram based CCVT energy solver. Lind *et al.* (2012) discussed the particle shifting and the problem they produces. All the methods are directly applied and served to fluid simulation without considering the sampling properties of the point distribution.

The standard SPH model discussed in Sec. 2.3 is used to demonstrate the fundamental sampling idea, and simplify the evaluation and exposition of our method, as it is the first application of fluid simulation to this problem domain. The SPH algorithm is naturally parallelized, ensuring the efficiency of SPH sampling. Additionally, the flexibility of SPH allows the method to be easily extended to general domain sampling

(including curves, surfaces and volumes), as well as supporting adaptive sampling and multi-class sampling. A new boundary correction method for SPH use is also proposed in sampling context, allowing the method for applications such as image stippling and re-meshing.

Alternative formulations of particle–based fluid, such as Solenthaler and Pajarola (2009); Ihmsen *et al.* (2014a); Macklin and Müller (2013), allow the simulation to run with larger time steps through greater stability but at a higher cost per step. While these methods have the potential to improve computational performance, they could also lead to slow convergence as the number of points increases since they all used Jacobi-style iterative methods (Macklin and Müller 2013). It is our intention to investigate the usage of these schemes as part of our future work.

7.6 Summary

In this chapter, the motivation of our SPH sampling is introduced, the related work on blue noise sampling is reviewed, and the evaluation methods for the blue noises are also demonstrated, as well as for the adaptive blue noise samplings. The next chapter will present the core idea and convergence criterion of the SPH sampling. The capabilities of creating controllable blue noise spectral profiles will also be shown in the next chapter.

Chapter 8

Density Difference and Sampling Analysis

8.1 The Core Idea

The concept of SPH in fluid simulation is to interpolate fluid quantities at arbitrary positions and to approximate the spatial derivatives with a finite number of sample positions (Ihmsen *et al.* 2014b). SPH is used to solve the Navier-Stokes equations. In the regions with a smooth flow field SPH particles are uniformly distributed (Adami *et al.* 2013) – which provides an intuition for its use in generating blue noise samples.

8.1.1 The Basic Algorithm

Following Eq. 2.19 in SPH and presuming each point has the same mass m, the point density can be represented by (Hu and Adams 2006; Tartakovsky *et al.* 2007):

$$\rho_i = m \sum_j W_{ij} \tag{8.1}$$

where W_{ij} is the abbreviation of $W(\mathbf{x}_i - \mathbf{x}_j, h)$. It is known that the

relationship between the mass the density can also be written as:

$$\rho_i = \frac{m}{V_i}. \tag{8.2}$$

where V_i is the volume. Notice that the volume of points will be the same if the density of points is the same.

According to the pressure force density in Eq. 2.23, the pressure force can be written as:

$$\mathbf{F}_i^{\text{pres}} = -m \sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W_{ij} \tag{8.3}$$

where pressure p is calculated as Eq. 2.24 from the difference between the actual density of the point and the rest density. The difference $\lambda = \rho - \rho_0$ is defined as the *density difference* which will be the key of the controllable blue noises.

The pressure force introduces a spring-like behaviour to the pressure calculation: a density higher than the rest density produces a positive pressure that pushes the points away; a lower density results in a negative pressure and brings the points together. Therefore, the pressure term aims to minimize the density differences throughout the fluid. For blue noise sampling, to avoid the attracting pressure forces the same concept of fluid simulation adopted — clamp the negative pressure to 0 in each iteration.

If a point is inserted at a density centre where there is no density difference, the pressure force returns to 0. This implies at equilibrium the pressure and density will be identical everywhere, thus the same volume (capacity) is achieved according to Eq. 8.2. This implicitly matches the capacity-constraint in CCVT which maintains the *equal area*, while SPH converges to *equal density*. This property of SPH sampling guarantees the quality of the generated blue noises which will automatically avoid distributions that are too regular.

When using SPH for fluid simulations a viscosity force is used to synchronize the velocity of all particles. In the sampling context the

smooth dynamic flow of a simulation is irrelevant: instead the samples are wanted to settle with 0 velocity as quickly as possible. Since SPH is a force driven solver, without viscosity points easily overshoot. In our method, a damping coefficient δ is applied to the velocity, achieving an equivalent behaviour to viscosity but with considerably better performance. The damping coefficient plays a similar role to the viscosity coefficient: it affects how quickly the particle reacts to the pressure. $\delta = 0.9$ is found works well in our experiments.

8.1.2 Convergence

The sampling is considered to have converged when the point displacement $\|\mathbf{x}' - \mathbf{x}\|$ is less than ϵ, where $\mathbf{x}'$ is the updated point position. ϵ is set to $0.01d$, where d is the average distance between adjacent samples which can be approximated by $d = \sqrt[n]{V/N}$ for n-dimensional sampling.

Upon convergence, the points have the following properties:

1. All points have approximately the same density, i.e. $\forall i, j,\ \rho_i \approx \rho_j$. Notice that this does *not* imply point density reaches the rest density.

2. The pressure force of all points reaches zero, i.e. $\forall i,\ \mathbf{F}_i^{\text{pres}} = 0$.

The first property means there is no density difference among points and the volume of the points are the same (surface area in 2D). Combining this with Eq. 8.3 and Eq. 2.24, under the setting of constant point mass, the second property can be written as:

$$\mathbf{F}_i^{\text{pres}} = 0 \approx -\frac{2m^2 k\left(\rho_i - \rho_0\right)}{\rho_i^2} \sum_j \nabla W_{ij} \tag{8.4}$$

The above equation implies that upon convergence, for any point i, one of the two following conditions must stand:

1. $\rho_i = \rho_0$;

2. $\sum_j \nabla W_{ij} = 0$.

When the first condition is satisfied for all points with a smooth kernel, it can be assumed that the density is approximately the same in the whole domain, or:

$$\forall \mathbf{x},\ m \sum_j W(\mathbf{x} - \mathbf{x}_j, h) \approx \rho_0 \tag{8.5}$$

Notice that in practice those two conditions are not strictly satisfied since in implementation the algorithm stops when point movement is small enough. Every point cannot be precisely controlled to converge with certain condition. There is a trade-off between these two conditions.

The different trade-offs can be achieved by varying the density difference λ. When the density is far away from the rest density, the points can only converge with the second condition. The condition of $\sum_j \nabla W_{ij} = 0$ actually asks for all the forces acting on a sample to be symmetrical, the points in this case will favour a regular distribution — hexagonal pattern. The results are very similar with the fully-converged Lloyd's relaxation, shown in Fig. 8.1 and Fig. 8.2.

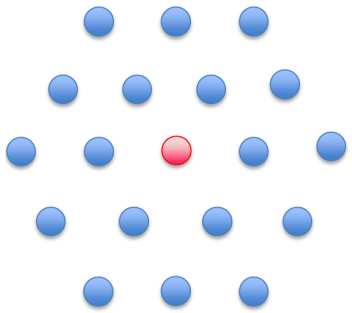


Figure 8.1: *Hexagonal pattern with* $\sum_j \nabla W_{ij} = 0$

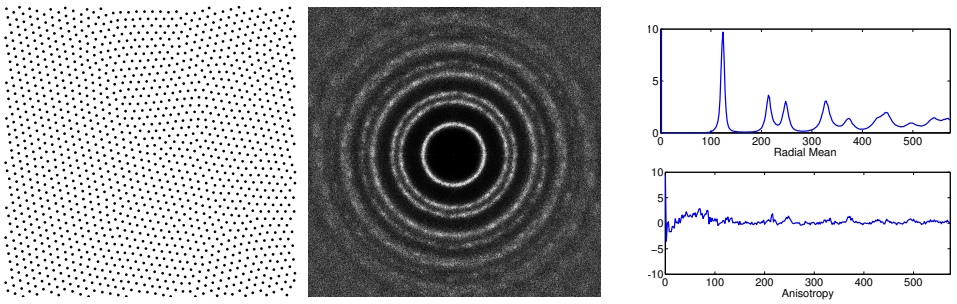


Figure 8.2: *Using our SPH sampling method to simulate the Lloyd's relaxation profile. This regular hexagonal distribution results from the convergence condition of* $\sum_j \nabla W_{ij} = 0$.

While if the rest density is set close to the actual density of each particle, the points will quickly converge with the first condition which allows more random distributions. This idea corresponds with CCVT method which has the volume constraint while ours have the same density. In this

case the resulting point distribution will be more likely to form tetragons, heptagons or pentagons — with similar results to CCVT — as demonstrated in Fig. 8.3 and Fig. 8.4. Later in Chap. 11, this sampling with CCVT profile will be used in the dissolution simulations.

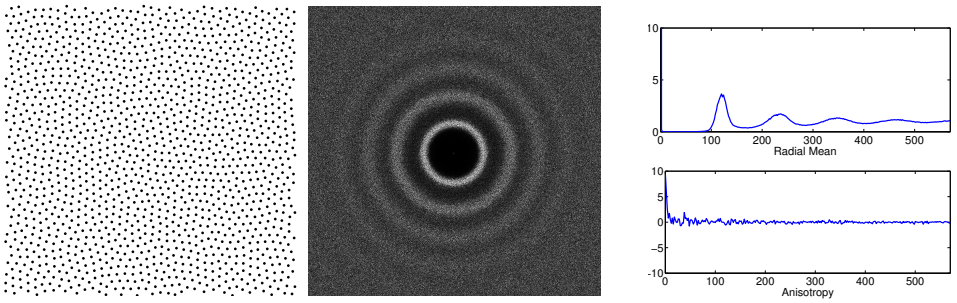


Figure 8.4: *Using our SPH sampling method to simulate the CCVT profile. This distribution results from the convergence condition of* $\rho_i = \rho_0$.

To avoid the regularity artefacts, most points are wanted to converge with the condition of $\rho_i = \rho_0$, rather than $\sum_j \nabla W_{ij} = 0$. In free-surface fluid simulation, the pressure of particles tends towards 0 when SPH reaches the stable state. In sampling context, the sampling area is pre-set and the user controls the number of the points, so the density is not guaranteed to equals to the initial rest density.

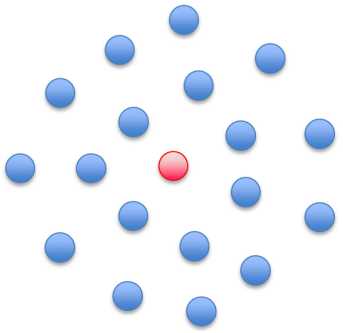


Figure 8.3: *CCVT profile with* $\rho_i = \rho_0$.

Therefore, different from the SPH in fluid simulation, the rest density is updated every time-step to meet the convergence criterion. To make the rest density close to the actual density of all the particles, the rest density is set as the average density $\bar{\rho}$ of all points at every frame:

$$\rho_0 = \rho_i \approx \bar{\rho} = \frac{1}{N} \sum_i \rho_i \tag{8.6}$$

For accuracy, only points not near the surface count into the average density since the density of surface points are deficient due to the lack

of neighbours.

If there is user defined density difference λ, the rest density is updated to keep the user defined density difference:

$$\rho_0 = \frac{1}{N}\sum_i \rho_i - \lambda \tag{8.7}$$

By tuning the density difference λ, our method can create the blue noises with different distributions between the trade-off of these two convergence criteria which will be detailed in Sec. 8.2.1.

Since negative pressure has the same effect as attractive force this can lead to sample clustering. To avoid this artefact in practice, the rest density should always be smaller than the average density, which means $\lambda > 0$ is always true.

Discussion. The choice of kernel function affects the accuracy of the summation (Ihmsen *et al.* 2014b). If not otherwise specified, the standard kernels discussed in Sec. 2.3.5 are used. The use of alternative kernels is discussed in Sec. 12.2. Note to ensure the normalization criterion the coefficients of the kernels in 2D are different from the ones in 3D.

8.2 Varying Blue Noises

The choices of SPH parameters and kernels directly affect the sampling patterns and distribution properties. Sec. 8.2.1 discusses how the density difference affects the trade-off of noise and aliasing of samplings. Later in Sec. 12.2 SPH kernel and its influence on the distribution properties of samples will be evaluated through experimentations.

8.2.1 Trade-off between Noise and Aliasing

Sec. 7.4 introduced two parameters which measured the sampling properties: effective Nyquist frequency ν_{eff} and oscillation Ω. Ideally the perfect blue noise shall have high ν_{eff} with low Ω. But unfortunately high ν_{eff}

always comes with high Ω, thus users have to make the trade-off between those two properties, or in other words, the trade-off between noise and aliasing. While most of the existing blue noise sampling methods are implicitly making such trade-offs, there are few discussions about how to have direct control over it. In our SPH sampling method, the trade-off between ν_{eff} – Ω can be controlled by varying the density difference parameter.

The trade-off between ν_{eff} – Ω has to do with the convergence conditions of our algorithm. As mentioned in Sec.8.1.2, there is a trade-off between the two convergence criteria of the algorithm. When ρ_0 is set closer to $\bar{\rho}$ upon convergence, the convergence condition of $\rho_i = \rho_0$ will be more strictly satisfied. In this case, as is shown in experiments (Fig. 8.4) samples will have lower Ω while compromising ν_{eff}.

When the density difference is large, more samples will converge with $\sum_j \nabla W_{ij} = 0$ and produce results with high ν_{eff}, at the cost of high Ω as in Fig. 8.2. Notice that those samples with hexagon patterns actually give the theoretical upper bound of effective Nyquist frequency (Dippé and Wold 1985).

One can easily control the trade-off between ν_{eff} and Ω by tuning the density difference, exposing a single user controlled parameter for this purpose. A lower difference refers to both low ν_{eff} and Ω, and a high difference gives samples with high ν_{eff} and Ω. The experimental results are shown in Tab. 8.1. By changing the density difference, a smooth transition can be seen on the Fourier power spectrum. The sampling method runs extremely fast which generates various blue noise with 10000 points per second.

To directly see how the density difference affects ν_{eff} and Ω, their relation is also plotted in Fig. 8.5: providing a guide to achieve preferred ν_{eff} and Ω. Fig. 8.6 plots the achievable ν_{eff} against the corresponding Ω alongside other sampling methods for comparison. This curve describes what are the ν_{eff} - Ω pairs that our sampling method can achieve. The curve of SPH sampling passes through Lloyd's relaxation and is close to the CCVT method, and can also give sampling patterns in a continuous

range between these two methods and even beyond them. Methods that are optimized for low oscillation with high effective Nyquist frequency, such as the step function and single-peak function (Heck *et al.* 2013) give a better trade-off than our method, although these methods are not directly tunable.

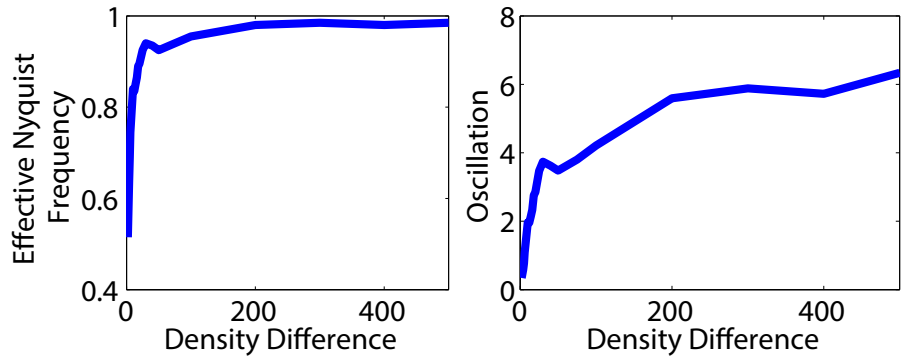


Figure 8.5: *Effective Nyquist frequency and oscillation changes as the 'density difference' parameter is increased.*

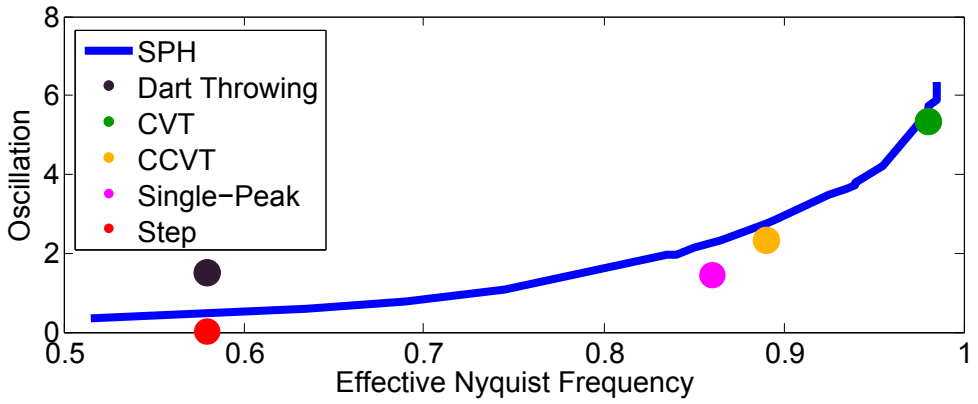


Figure 8.6: *Trade-off between Effective Nyquist Frequency and oscillation. The blue line shows the range that our SPH sampling method can achieve, by varying the 'density difference' parameter. It is compared with other methods by plotting their trade-off in the same figure. Note that most of them can only generate one distribution profile, therefore are represented by single dots. The data source of these methods is from Heck* et al. *(2013).*

The technique by Chen *et al.* (2012) also provides trade-offs of the cut-off frequency and undulation, allowing intermediate results between CVT and CCVT. However their range of trade-off is limited by the CCVT

pattern, which corresponds to the (cut-off frequency) region of 0.89–0.98 in Fig. 8.6. Our method, on the other hand, can go way beyond the CCVT pattern and provides a much wider range of sampling patterns, which allows for bigger trade-off area of 0.52–0.98.

As discussed above, the density difference significantly effects the sampling quality. If one wants a specific trade-off between ν_{eff} and Ω, it can be easily achieved by looking up the corresponding density difference value. For convenience, the density difference parameter is plotted against effective Nyquist frequency of the 2D domain, and fit it with a curve using the regression function $f(x) = ae^{bx} + ce^{dx}$, with parameters $a = 15.28, b = 1.154, c = 0.2565, d = 7.729$, as is shown in Fig. 8.7. To achieve a specific effective Nyquist frequency ν_{eff}, one just needs to set the corresponding density difference $\lambda = f(\nu_{\text{eff}})$.

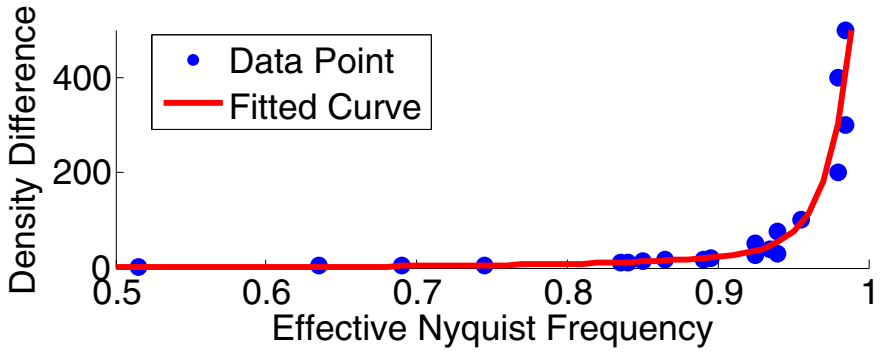


Figure 8.7: *Plot of density difference against effective Nyquist frequency and the fitted curve in 2D domain.*

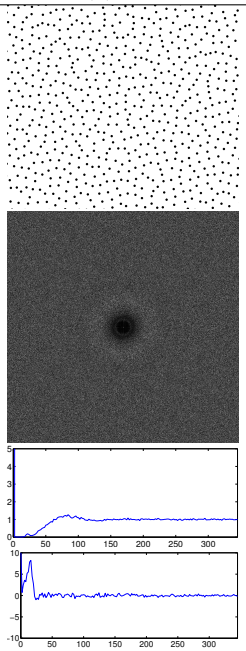
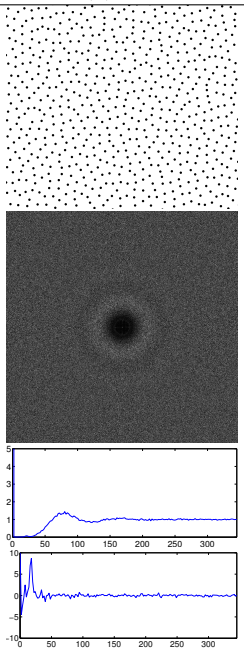
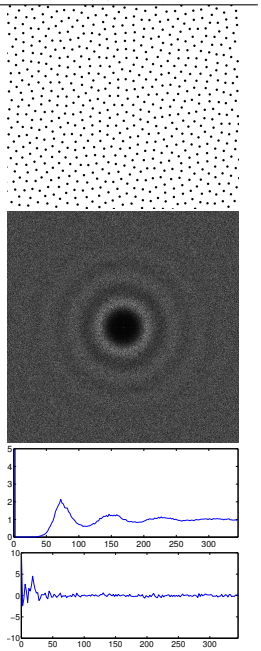
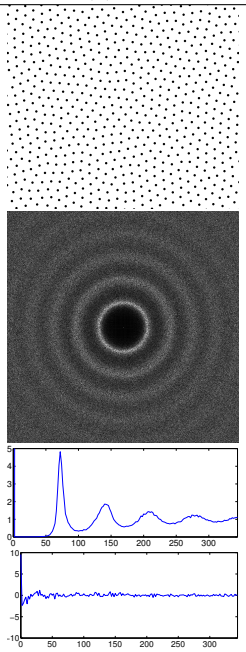
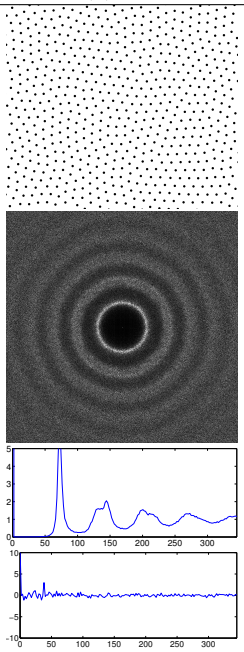
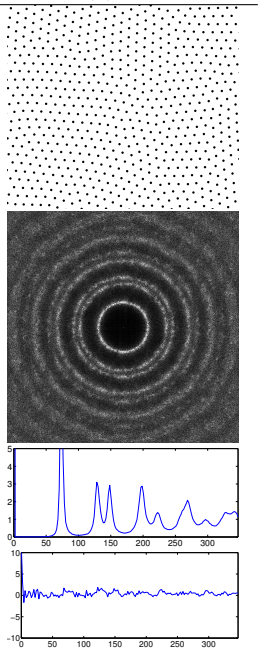
	Density Diff.: 2 Iterations: 373 Time (s): 1.108	Density Diff.: 4 Iterations: 344 Time (s): 0.936	Density Diff.: 8 Iterations: 300 Time (s): 0.842
points spec- trum			
	Density Diff. : 25 Iterations: 288 Time (s) : 0.780	Density Diff.: 100 Iterations: 217 Time (s) : 0.593	Density Diff.:500 Iterations: 228 Time (s) : 0.640
points spec- trum			

Table 8.1: *This table shows samples generate using our algorithm, by varying the 'density difference' parameter. Each example shows Fourier power spectrum, associated radial means and anisotropy of the spectrum. With density difference from small to large, the spectrum transitions from low effective Nyquist Frequency and oscillation to high effective Nyquist Frequency and oscillation.*

8.3 Summary

In this chapter, the similarities and differences between SPH sampling and other popular blue noise sampling methods are analysed. Convergence criterion of SPH sampling is explained. By adjusting the density difference a variety of blue noise sample patterns are able to be generated with different distribution properties that range from Lloyd's relaxation to CCVT, and even beyond of these. The trade-offs between effective Nyquist frequency and oscillation in the Fourier power spectrum is also demonstrated. By doing a simple table lookup people can generate samples with desired blue noise properties.

The implementation of SPH in the area of sampling is not straightforward. Next chapter will discuss the SPH sampling algorithm in general dimentions as well as the adaptive sampling.

Chapter 9

SPH Sampling Algorithm

In the last chapter, the reason of using the fluid methods in the sampling problems is disclosed. From the analysis of convergence criterion, it is found that the density difference greatly affected the distribution of blue noises. By tuning the density difference, a continuous trade-off between the effective Nyquist frequency and oscillation of blue noises were able to be achieved. It is also experimentally proved that the different kernels also affected the sampling patterns. This chapter will mainly describe the SPH sampling algorithm with our novel boundary correction method (Sec. 9.1), as well as the implementation of surface sampling(Sec. 9.2), adaptive sampling (Sec. 9.3) and multi-class sampling (Sec. 9.4).

9.1 Volume and Surface Sampling

The points are initialized as white noise within a given boundary using the same Signed-Distance Field (SDF) method discussed in Sec. 5.2. Each sample is modelled as an SPH particle, which carries associated properties, e.g. position, velocity, density and pressure. SPH is run over all the points, with the acceleration of the points obeying Newton's law of motion $\mathbf{a}_i = \mathbf{F}_i^{\text{pres}}/m_i$. Fig. 9.1 shows the SPH sampling process within a 2D bunny shape. The rightmost image shows the converged result. Note that in the sampling context, gravity is not needed, and the viscosity is

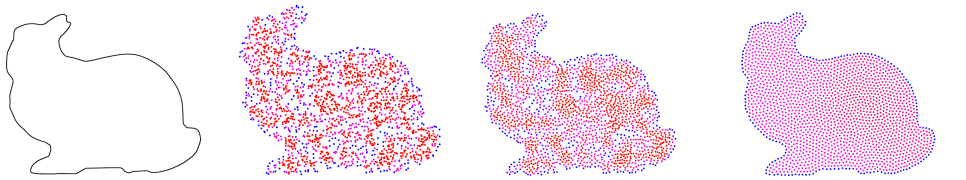


Figure 9.1: *Sampling the interior and boundary of a 2D bunny shape using our SPH-based method. The color visualizes the density value of each point, from high (red) to low (blue). Note the distribution quality of the resulting samples.*

replaced by a damping coefficient δ.

In SPH, points are relocated to achieve equivalent density in the sampling region. However, due to the insufficient sampling along the object boundary, the points on the boundary become highly disordered, shown in Fig 9.3(a). Therefore, a correction-cohesion model is proposed for the boundary correction.

9.1.1 Correction Force

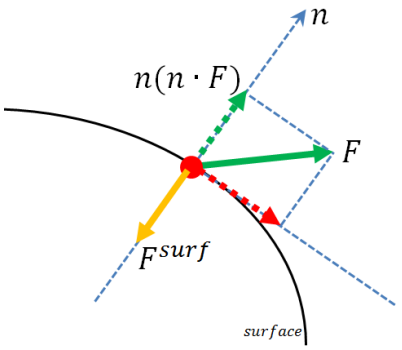


Figure 9.2: *The correction force negates force components in the normal direction to the surface, ensuring that points on the boundary move only along the boundary.*

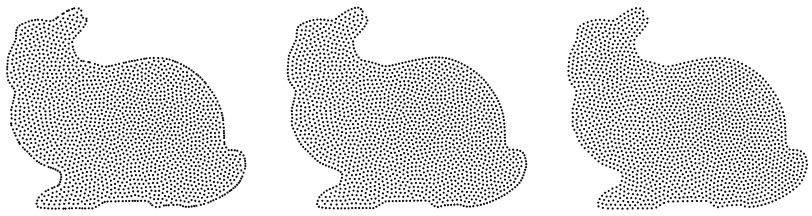
The net forces of the samples on the boundary point in the direction of the boundary normal. Therefore a correction force is needed which allows the boundary samples to relax onto the boundary without being pushed outwards. To allow samples to move freely along the boundary,

the force in the normal direction can be counteracted and only the force along the tangential direction be kept:

$$\mathbf{F}_i^{\text{surf}} = -\mathbf{n}(\mathbf{n} \cdot \mathbf{F}_i^{\text{pres}}) \tag{9.1}$$

This correction force ensures that the samples near the boundary remain on the boundary of the object, resulting in a better quality of the surface sampling. The same idea is applied to the velocity when a point hits the boundary: only the tangential component of the velocity is kept while the perpendicular component is set to 0, which also improves stability. The particle distribution with correction force is shown in Fig. 9.3(b).

Due to the insufficient sampling around the boundary, density changes dramatically for points on the boundary. Therefore the normal of the surface samples can be calculated from the gradient of the density using Eq. 2.28. The resulting normal points from the high density to the low density, out the volume.



(a) No correction (b) +corrective force (c) +cohesion forces

Figure 9.3: *Boundary corrections of SPH sampling. In (a) the boundary points are irregularly spaced after sampling. In (b), the boundary points are more relaxed after adding the correction force, but the sampling rate is much denser than the interior. In (c) both correction and cohesion forces are applied, and the resulting boundary samples are coherent with the interior samples.*

9.1.2 Cohesion Force

Since the density of boundary points is not adjusted, the boundary will attract more points to compensate the density loss, leading to a more

dense sampling (shown in Fig. 9.3(b)). With only the correction force, once points are on the surface they cannot leave — potentially causing discontinuities between surface points and interior points. For this reason an additional force is introduced which removes the density difference between surface points and interior points.

The method of Akinci *et al.* (2013) is used which contains cohesion and surface area minimization terms. As our sampling model defines the surface area beforehand (unlike free–surface fluid), the surface area minimization term does not affect our results. Only the cohesion force is applied:

$$\mathbf{F}_i^{\text{cohe}} = -\gamma m_i m_j K_{ij} W_{cohe}(|\mathbf{r}_i - \mathbf{r}_j|) \frac{\mathbf{r}_i - \mathbf{r}_j}{|\mathbf{r}_i - \mathbf{r}_j|} \tag{9.2}$$

where γ is surface tension coefficient, W_{cohe} is the spline function defined by Akinci *et al.* (2013), and K_{ij} is a symmetric correction factor defined as $K_{ij} = \frac{2\rho_0}{\rho_i + \rho_j}$.

This cohesion force is applied to both interior and surface points. For interior points the cohesion force is almost the same everywhere as it converges. Surface points with smaller density have larger K_{ij} due to the lack of neighbouring samples, which amplifies the cohesion forces of boundary points. The effect of this force is demonstrated in Fig. 9.3(c). Note also that the corrected surface sampling refines the points near the boundary which have boundary points as neighbours.

The full method of volume and surface sampling is provided in Algorithm 2.

Discussion. Our method for surface and volume sampling ensures that the surface points are exactly on the surface and also guarantees the consistency between the surface and volume sampling. Most boundary correction methods in SPH do not guarantee the particles to converge exactly on a fixed boundary, e.g., Akinci *et al.* (2012, 2013). Therefore,the correction force here is necessary.

Input: Initial Random Sample Set P_o, Boundary Level set L, Sampling number N, count $i = 0$

Output: Blue noise samples P

while *not converged* **do**
  for $i = 0$ ***to*** N **do**
    calculate ρ using Eq. 8.1;
  end
  $\rho_0 \leftarrow$ set rest density using Eq. 8.7 according to the chosen density difference λ
  for $i = 0$ ***to*** N **do**
    $\mathbf{x}, \mathbf{v} \leftarrow$ the position and velocity of current point
    $\mathbf{F}^{\text{pres}} \leftarrow$ calculate pressure force using Eq. 8.3
    $\mathbf{F}^{\text{cohe}} \leftarrow$ calculate cohesion force using Eq. 9.2
    $\mathbf{n} \leftarrow$ calculate normal using Eq. 2.28
    if $\mathbf{x}$ *is on the surface* **then**
      $\mathbf{F}^{\text{surf}} \leftarrow$ calculate correction force using Eq. 9.1
      $\mathbf{a} \leftarrow \mathbf{F}^{\text{pres}} + \mathbf{F}^{\text{cohe}} + \mathbf{F}^{\text{surf}}$
    else
      $\mathbf{a} \leftarrow \mathbf{F}^{\text{pres}} + \mathbf{F}^{\text{cohe}}$
    end
    Integrate $\mathbf{a}$ to get velocity $\mathbf{v}'$ and position $\mathbf{x}'$ using Eq. 2.36 and Eq. 2.37
    Apply velocity damping $\mathbf{v}' \leftarrow \delta\mathbf{v}'$ instead of viscosity
    if $\|\mathbf{x}' - \mathbf{x}\| > \epsilon$ **then**
      if $\mathbf{x}'$ *is within* L **then**
        $\mathbf{x} \leftarrow \mathbf{x}'$
        $\mathbf{v} \leftarrow \mathbf{v}'$
      else
        $\mathbf{v} \leftarrow \mathbf{v}' - \mathbf{n}(\mathbf{n} \cdot \mathbf{v}')$
      end
    end
  end
end

Algorithm 2: *The SPH sampling algorithm for interior and boundary points.*

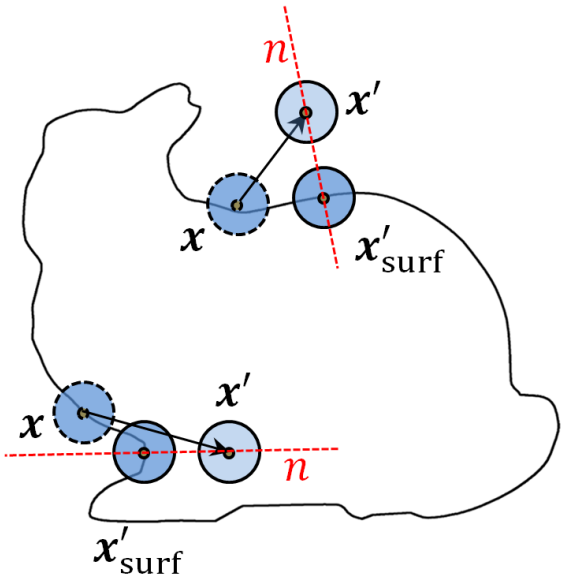


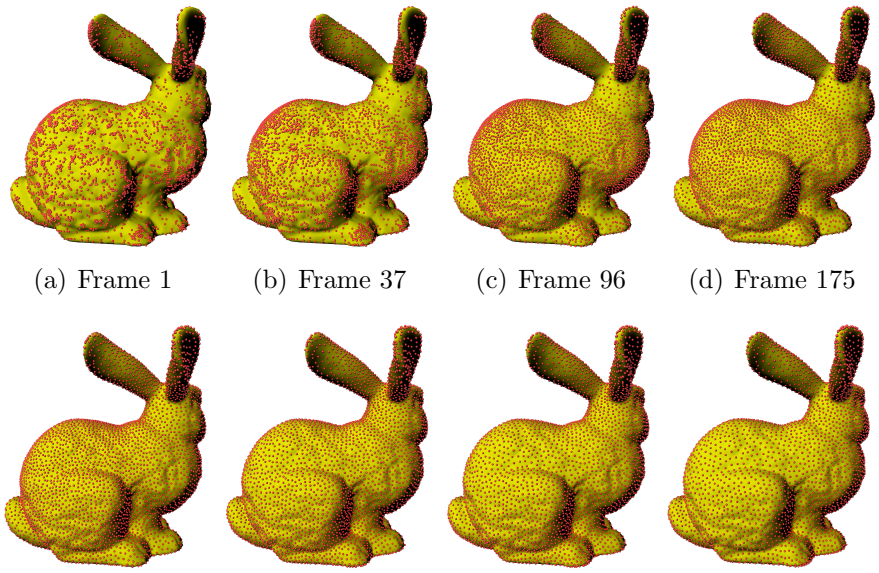
Figure 9.4: *The diagram of surface sampling algorithm. After point located at* $\boldsymbol{x}$ *is updated to* $\boldsymbol{x}'$ *according to pressure force and correction force, it is then mapped back to the surface at position* $\boldsymbol{x}'_{surf}$ *along the surface normal* $\boldsymbol{n}$*.*

9.2 Surface Sampling

Our method is also suitable for surface sampling without the interior points. It is achieved by only using the correction force and replacing the distance metric with geodesic distance.

Samples are initialized on the surface. Then SPH is run for all surface samples, calculating $\mathbf{F}^{\text{pres}}$ and $\mathbf{F}^{\text{surf}}$ for each sample. Instead of restricting the points within the object, the points are constrained to only move on the surface. Therefore within every time step of the SPH process the samples need to be mapped back onto the surface. The diagram of the algorithm is shown in Fig. 9.4. The position of the sample is firstly updated to position $\mathbf{x}'$ according to the acceleration calculated from the pressure and correction force. Then, the sample is mapped to the position $\mathbf{x}'_{surf}$ along the surface normal. Fig. 9.5 shows the sampling process on a bunny model.

In SPH, the Euclidean distance metric is used, which, for surface sampling, will lead to an undesirable distribution, especially for those



(a) Frame 1 (b) Frame 37 (c) Frame 96 (d) Frame 175

(e) Frame 275 (f) Frame 458 (g) Frame 609 (h) Frame 1049

Figure 9.5: *Surface sampling process of a bunny model.*

complex shapes with thin features. In surface sampling, geodesic distance needs to be used which takes the surface into account instead of Euclidean distance which measures the shortest distance in 3D space.

The accurate calculations of geodesic distance are often expensive and difficult to parallelize (Sethian 1995; Surazhsky *et al.* 2005). Ideally a simple approximation is wanted that only replies on the points and not on the mesh connectivity and parametrization since the surface is represented as point cloud. For this purpose, the light-weight algorithm of Bowers *et al.* (2010) is applied, which approximates the geodesic distance using normal information.

Assume there is a smooth curve on the surface which passes through two points p_1 and p_2 with respective normal $\mathbf{n}_1$ and $\mathbf{n}_2$, shown as the orange curve in Fig. 9.6. Since there is no existing density insufficient for the boundary points, SDF method is used to calculate the normals (Bærentzen and Aanæs 2002; Baerentzen and Aanaes 2005). The geodesic distance d_g is estimated as the length of the curve. The Euclidean distance between the two points is $d_e = ||p_2 - p_1||$ and the normalized vector connecting the two points is $\mathbf{u} = (p_2 - p_1)/d_e$. The cosine angles of the two normals with the direction $\mathbf{u}$ are $c_1 = \mathbf{n}_1 \cdot \mathbf{u}$ and

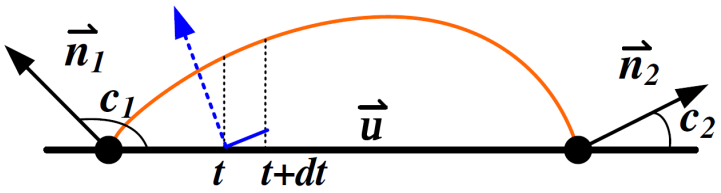


Figure 9.6: *Geodesic distance approximation. The blue line indicates the differential curve length at integration step t (reproduced from Bowers* et al. *(2010))*

$c_2 = \mathbf{n}_2 \cdot \mathbf{u}$.

However, explicitly constructing this curve would be complicated, since $\mathbf{n}_1$, $\mathbf{n}_2$, and $\mathbf{u}$ may not be co-planar. Bowers *et al.* (2010) uses differential steps along $\mathbf{u}$ to estimate d_g. At each step t, the cosine angle between the curve and $\mathbf{u}$ can be simply estimated as a linear interpolation between c_1 and c_2: $c(t) = (1-t)c_1 + tc_2$. The differential curve length along the tangent direction then can be estimated using $c(t)$. The geodesic distance d_g between p_1 and p_2 can be obtained by integrating the differential curve length as:

$$d_g = \int_0^1 \frac{d_e}{\sqrt{1-[(1-t)c_1 + tc_2]^2}} \mathrm{d}t = \frac{\arcsin c_1 - \arcsin c_2}{c_1 - c_2} d_e \qquad (9.3)$$

where if $c_1 = c_2$, the formula reduces to $\frac{d_e}{\sqrt{1-c_1^2}}$ using L'Hospital rule (Taylor 1952).

The main advantage of this approximation is that it is fast to compute and can easily fit into the existing algorithm. On the other hand, since it completely ignores the underlying mesh, it is only accurate when the surface changes smoothly between p_1 and p_2. Note, the geodesic distance is only needed to compute when $d_e < h$, because $d_g \geq d_e$. $d_e \geq h$ means $d_g \geq h$ while the geodesic distance is larger than the neighbourhood size. Usually h is small enough, thus the two points must be quite close spatially and the surface typically does not contain high-frequency changes at this scale level. Therefore in practice it is found that this approximation worked well for our method and it is unnecessary for us

Input: Initial Random Sample Set P_o, Boundary Level set L, Sampling number N, count $i = 0$
Output: Blue noise samples P
while *not converged* **do**
  for $i = 0$ ***to*** N **do**
    calculate ρ using Eq. 8.1;
  end
  $\rho_0 \leftarrow$ set rest density using Eq. 8.7 according to the chosen density difference λ
  for $i = 0$ ***to*** N **do**
    $\mathbf{x}, \mathbf{v} \leftarrow$ the position and velocity of current point
    $\mathbf{n} \leftarrow$ calculate normal using signed distance field
    $d_g \leftarrow$Calculate geodesic distances between points using Eq. 9.3
    $\mathbf{F}^{\text{pres}} \leftarrow$ calculate pressure force using Eq. 8.3
    $\mathbf{F}^{\text{surf}} \leftarrow$ calculate correction force using Eq. 9.1
    $\mathbf{a} \leftarrow \mathbf{F}^{\text{pres}} + \mathbf{F}^{\text{surf}}$
    Integrate $\mathbf{a}$ to get position $\mathbf{x}'$ and velocity $\mathbf{v}'$ using Eq. 2.36 and Eq. 2.37
    Apply velocity damping $\mathbf{v}' \leftarrow \delta\mathbf{v}'$ instead of viscosity
    Map $\mathbf{x}'$ to the surface position $\mathbf{x}'_{\text{surf}}$
    if $\|\mathbf{x}'_{\text{surf}} - \mathbf{x}\| > \epsilon$ **then**
      $\mathbf{x} \leftarrow \mathbf{x}'_{\text{surf}}$
      $\mathbf{v} \leftarrow \mathbf{v}'$
    end
  end
end

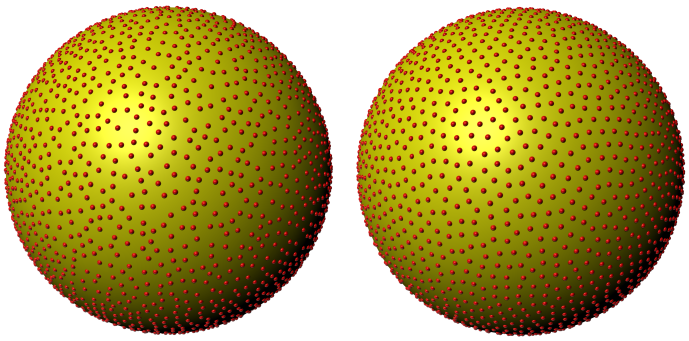
Algorithm 3: *The SPH sampling algorithm for surfaces.*

to have an accurate geodesic distance calculation.

The full algorithm is shown in Algorithm 3.

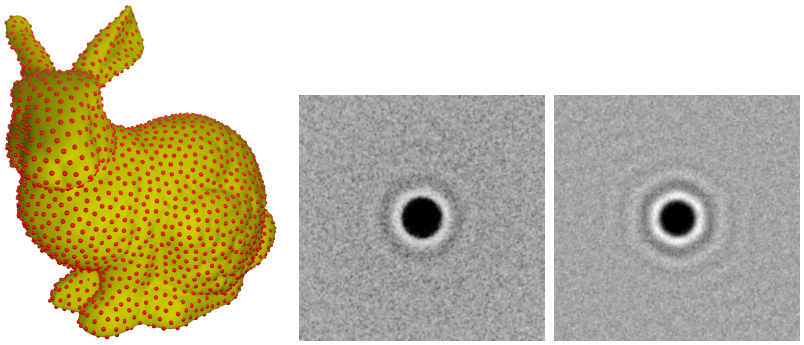
As discussed in Sec. 8.2.1, our algorithm can also generate surface samples with a variety of blue noise properties, including distributions similar to CCVT. Fig. 9.7 compares our surface sampling results with Poisson disk sampling of Bowers *et al.* (2010) on a sphere. Our samples exhibit uniform distributions similar to the CCVT profile.

Fig. 9.8 demonstrates a surface sampling on a bunny model and compare the Differential Domain Function (DDF) (Wei and Wang 2011) with that of 2D CCVT (Balzer *et al.* 2009). The results look qualitatively similar – the difference is mainly due to the geodesic distance approximation when evaluating surface DDF.



(a) Bowers *et al.* (2010) (b) Ours with CCVT profile

Figure 9.7: *Comparison of Poisson disk surface sampling (Bowers* et al. *(2010)) with our method.*



(a) (b) (c)

Figure 9.8: *Surface sampling of a bunny model. (a) visualizes the samples. (b) shows the Differential Domain Function (DDF) of the bunny surface samples (averaged over 10 runs) and (c) shows the DDF of the CCVT method (Balzer* et al. *2009), both calculated using the method of Wei and Wang (2011).*

Our surface sampling method can also be implemented to sample a curve, which is often neglected by most sampling methods. The curve can be important to frame the boundary shape. Fig. 9.9 demonstrates our sampling algorithm applied to a feature curve of a bowl in 1D.

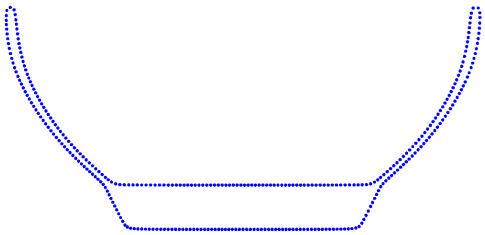


Figure 9.9: *1D sampling portraits the main curve of a bowl.*

Another interesting property of our algorithm is that samples implicitly preserve the mesh features such as ridges and valleys. This is because in high curvature regions, points will be automatically pushed along the tangent direction towards high curvature region, shown as the red points in Fig. 9.10. This effect is actually beneficial for many sampling applications such as rendering and remeshing, capturing high frequency features on the mesh. Its application will be discussed in Sec. 10.2.

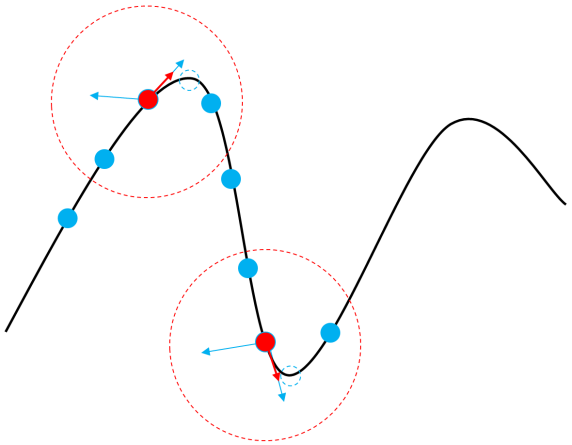


Figure 9.10: *Surface sampling property. Our algorithm preserves the mesh features by pushing the particles towards high curvature regions. The red arrows show the summary forces of the pressure forces (blue arrows) from the neighbouring particles. The natural equilibrium of this configuration is when the red particle rests at the feature of high curvature.*

9.3 Adaptive Sampling

Adaptive sampling is achieved by applying a distance field scale $s(\mathbf{x})$ to the sample properties. As in previous work (Wei and Wang 2011; Zhou *et al.* 2012), the distance field can be calculated from the intensity image $I(\mathbf{x})$ as $s(\mathbf{x}) = \frac{1}{\sqrt{I(\mathbf{x})}}$, or be defined as the inverse of the size function. Fig. 9.11 shows two examples of adaptively sampled 2D bunny. One is sampled according to the density function of an image pattern and the other is distributed in terms of its feature size.

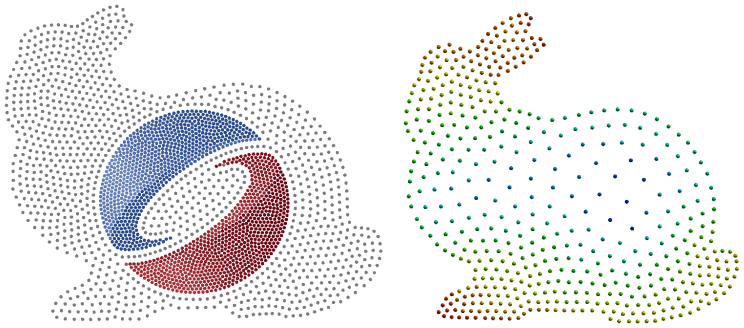


Figure 9.11: *2D bunny adaptively sampled according to density function and feature size.*

The unwrapped distance between two points is calculated as:

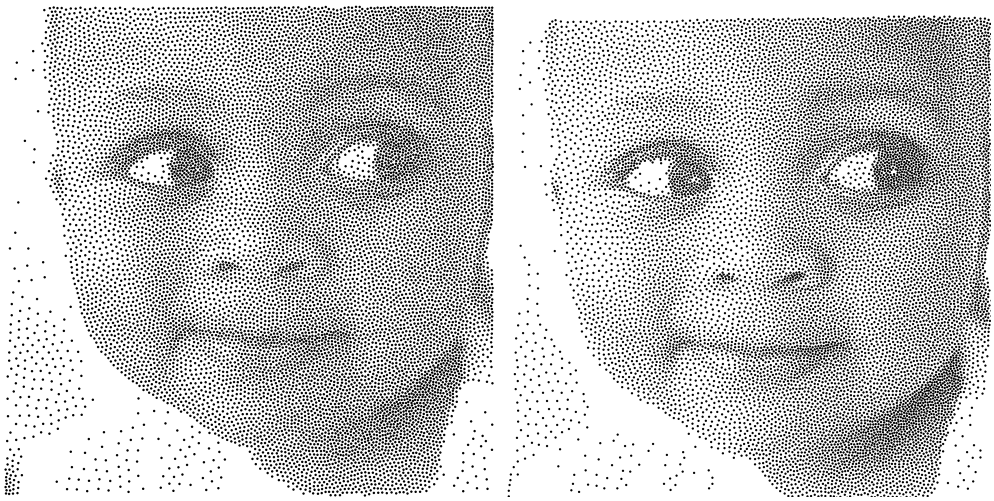
$$\tilde{s}(\mathbf{x}_i, \mathbf{x}_j) = \frac{2(\mathbf{x}_i - \mathbf{x}_j)}{s(\mathbf{x}_i) + s(\mathbf{x}_j)} \tag{9.4}$$

Which is then used in the kernel functions:

$$W_{ij} = W(\tilde{s}(\mathbf{x}_i, \mathbf{x}_j), h) \tag{9.5}$$

As the scale is increased, the kernels will cover larger regions. In equilibrium, to keep a similar density ρ as defined in Eq. 8.1, the total number of points in the large kernel region shall remain similar as those with smaller scale. As a result, points will become sparser in high scale regions, and denser in low scale regions. The update of the sample position therefore needs to correspond to the distance field scale as well:

$$(\mathbf{x} - \mathbf{x}') \leftarrow \tilde{s}(\mathbf{x})(\mathbf{x} - \mathbf{x}'). \tag{9.6}$$



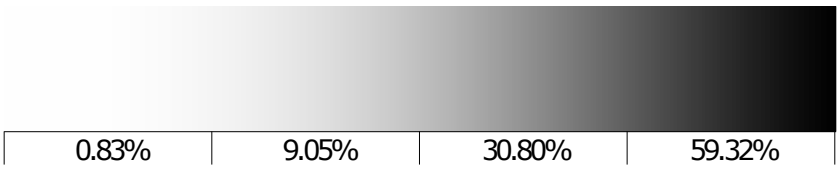
(a) Fattal (2011) (b) Our method

Figure 9.12: *Image stippling comparison. The left image is the result of Fattal (2011) and the right results from our method. Both images contain 13000 points.*

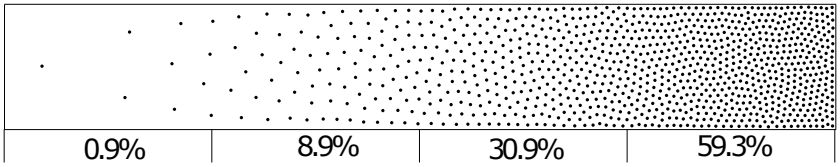
Notice that when computing the convergence criterion, one still needs to use the unscaled distance instead of the new one.

Image stippling is a common problem in Computer Graphics. Fig. 9.12 compares our method with the work of Fattal (2011). The results show that there is less definition around boundary while using SPH sampling, but more contrast in the filled region.

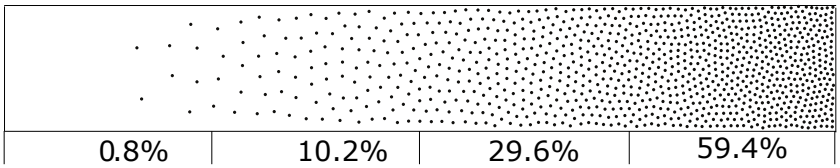
The quality of adaptive sampling is additionally compared in Fig. 9.13 with Balzer *et al.* (2009) and Fattal (2011) using a quadratic ramp as the scale function. In the coarse region, our method does not produce as good results as in the dense region. It is because the choice of h limits the influence range. If h is too large, it will cause the points to blur the given density function.



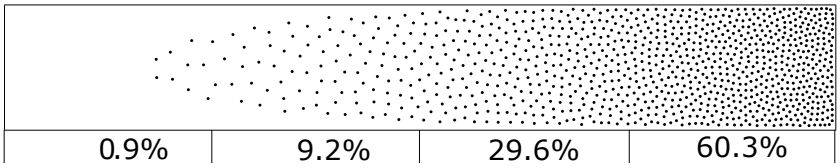
(a) Quadratic density ramp



(b) Balzer *et al.* (2009)



(c) Fattal (2011)



(d) Our method

Figure 9.13: *Comparison of adaptive sampling of our method vs. Balzer* et al. *(2009) and Fattal (2011). Here the spatial density function is defined using the quadratic ramp as show on the top. Every example contains 1000 points, the percentage underneath indicate the density ratios for each quarto of the ramp.*

9.4 Multi-class Sampling

Multi-class sampling is a property of a labelled set of samples where each class and the combination of classes exhibit desirable noise properties. It has applications for phenomena requiring multiple classes of samples, such as object distribution (Cohen *et al.* 2003), color stippling (Kopf *et al.* 2006) and color sensor layout (Ben-Ezra *et al.* 2007).

The natural property of mixing fluid using SPH gives rise to the application of multi–class noise sampling similar to Wei (2010). In his paper, Wei defined two methods for multi-class sampling — soft disk sampling and hard disk sampling. Hard disk sampling could not control the number of points for each class while soft disk sampling avoided this problem, it was yet much more expensive to ensure the sample uniformity. In contrast, our method can accurately control the number for each class without adding computational cost to our SPH sampling method of single class.

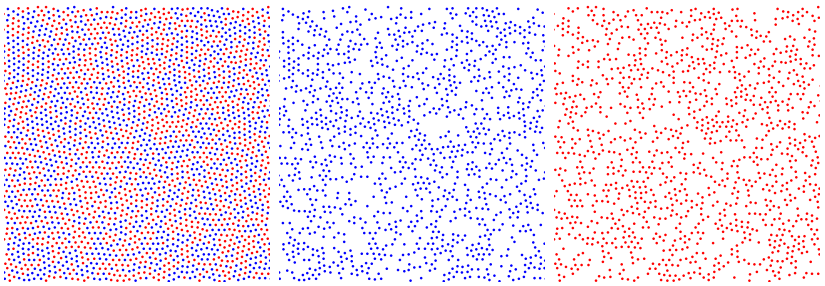


Figure 9.14: *Multi–class sampling without different class handling fails to reproduce a blue noise profile in each class.*

Multi-class blue noise sampling is achieved by giving different points a different class ID $\{c_i\}_{i=0:C-1}$ for C classes of points. A number of samples for each class is initially specified. If SPH is run for all the samples without any special handling, the total set will be uniform blue noise, but not for the individual class (as shown in Fig. 9.14).

Take two classes sampling as an example, if the same class is wanted to be uniformly sampled, the distance between samples in the same class r_1 and r_2 in 2D should satisfy $r_1 = \sqrt{3}r_2$ in an ideal hexagon pattern,

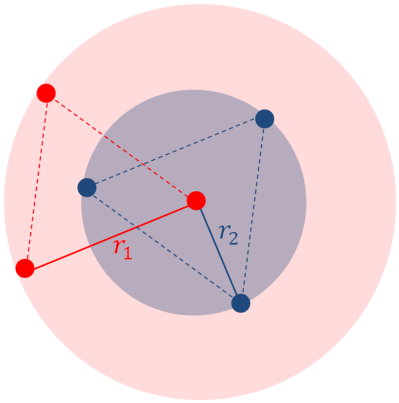


Figure 9.15: *Derivation of the scaling factors for multi–class sampling.*

shown in Fig. 9.15. This implies that multi-class sampling can be simply achieved by modifying the concept of adaptive sampling. The distance field scale for samples from the same classes is set as s_1 (corresponding to the $s(\mathbf{x})$ in adaptive sampling), the distance field scale for samples from different classes s_2 has to fulfil $3s_1 = s_2$ in order to achieve $r_1 = \sqrt{3}r_2$. The results are shown in Fig. 9.16. It should be noted that while Wei (2010) produces results that are Poisson-disk sampling, our method gives results closer to those of CCVT. Fig. 9.17 shows the two multi-class sampling of Wei (2010).

The proposed SPH sampling method works for multi–classes without the need of building a complex matrix for multi–class as in Wei (2010). The computational cost of multi-classes is almost the same with one class. A five classes blue noise sampling is shown in Fig. 9.18. It is achieved easily by initializing five classes of samples and executing the ratio discussed above for the samples in or not in the same class.

9.5 Limitation of SPH Sampling

According to the Nyquist–Shannon theorem (Jerri 1977), the sufficient sampling rate is twice the bandwidth of a band–limited function, so the relationship between smooth length h and average adjacent distance d can be written as $h \geq 2d$. Therefore, h should not be too small. Otherwise, there will be insufficient neighbour particles for SPH to run.

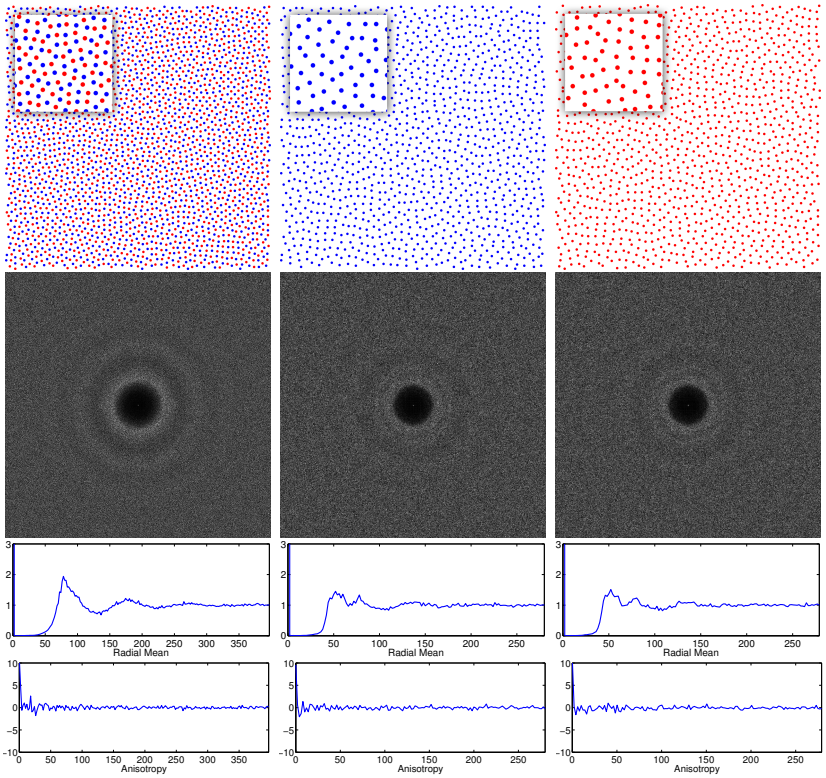


Figure 9.16: *Our Multi-class sampling. SPH sampling produces samples which exhibits blue noise distribution for each class as well as the total set. See Sec. 7.4 about the discussion of blue noise quality, and the resulting blue noises are close to the step blue noise profile.*

However if the object has features which are small enough, our sampling method may fail if h is chosen too large. Fig. 9.19 demonstrates the case of an object with an internal boundary. The smooth length must be chosen to be smaller than the minimum feature size, otherwise the sampling on the boundary will fail.

For adaptive sampling, if the smooth length is too large, the density function will be blurred as discussed with Fig. 9.13.

9.6 Summary

This chapter focuses on the SPH sampling algorithm and the differences between the SPH methods used in fluid simulation and in sampling area. A new boundary correction method is proposed to solve

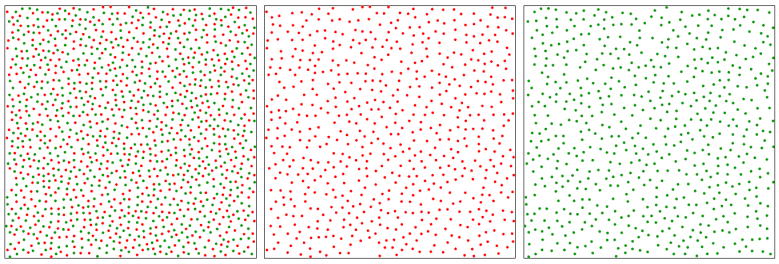


Figure 9.17: *Multi-class sampling of Wei (2010).*

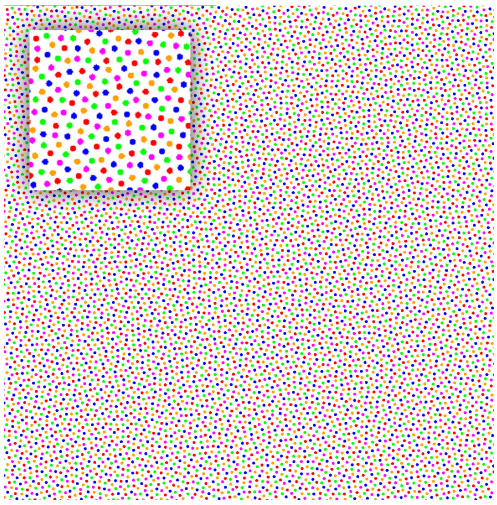


Figure 9.18: *SPH blue noise sampling of five classes.*

boundary deficiency of SPH in the sampling area. The flexibility of SPH allows spatially-varying point density, leading to adaptive sampling. Our method supports sampling in general dimensions as well as the multi-class blue noise sampling. Moreover, SPH sampling is easy to implement in parallel according to Chap. 3 which ensures the efficiency of computation. The applications of SPH sampling and its performance will be discussed in the next chapter.

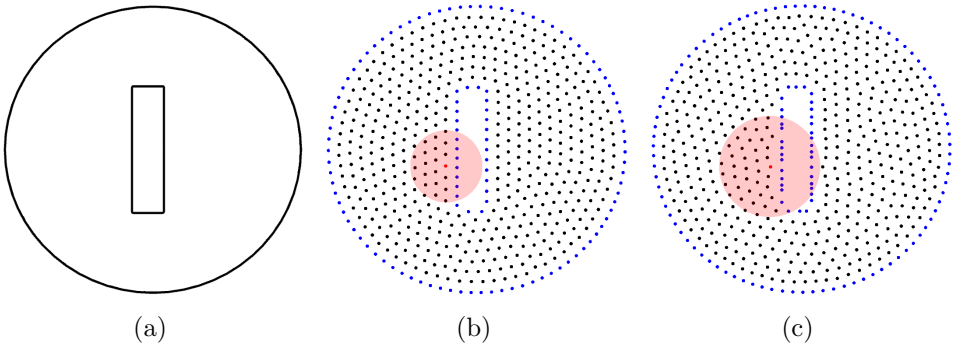


Figure 9.19: *Choice of smooth length. If the smooth length is smaller than the feature size (b), the interior boundary will be correctly sampled. Otherwise the sampling on the interior boundary will be incorrect (c).*

Chapter 10

Applications of SPH Sampling

In the preceding chapter, the SPH sampling algorithms have been demonstrated as well as the surface, adaptive and multi-class sampling. This chapter will mainly focus on the performance of our SPH sampling and other applications such as remeshing and adaptive volume sampling.

10.1 Performance

Our SPH implementation is based on Chapter 3. Neighbour search method discussed in Chapter 3.1 is used, and further acceleration can be achieved with more advanced data structures. Currently our method performs slower than Wachtel *et al.* (2014), but their method is difficult to extend to general sampling domains.

The computational cost of our algorithm depends on the number of samples. Specifically, the time complexity of our method is $O(N)$ for each iteration with uniform grid structure on the CPU (where N is the number of samples), while the complexity of the Lloyd's relaxation method is $O(N \log N)$ (Du and Emelianenko 2006). Table. 10.1 shows the computation times of our method for generating 16384 samples with Lloyd's and CCVT profiles respectively, and compare them with the previous

work.

	Lloyd's profile	*CCVT profile*
Fast CCVT Li *et al.* (2010a)	-	35.75s
Lloyd's relaxation	37.38s	-
Our method	0.687s	0.952s

Table 10.1: *Performance of sampling with CCVT and Lloyd's profile. Comparison data originates from Li* et al. *(2010a)*

Other supported features of our method, such as surface and adaptive sampling, incur additional computational costs. Table. 10.2 shows the computation times of different examples presented in this chapter.

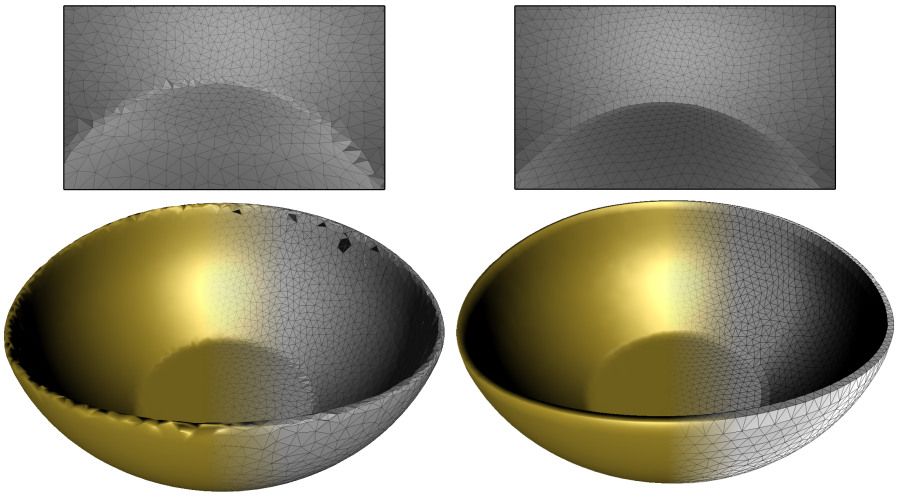
Example	*Samples*	*Iterations*	*Time(s)*
Fig. 9.1	2000	49	2.04
Fig. 9.7	3000	94	2.85
Fig. 9.8	2500	90	4.79
Fig. 9.12	13000	159	2.93
Fig. 9.16	10000	312	4.77
Fig. 10.1	5000	223	6.71
Fig. 10.2	10000	445	9.10

Table 10.2: *The performance of SPH sampling in different examples.*

10.2 Remeshing

Our surface sampling algorithm enables us to achieve excellent surface sampling with accurate control of the sampling number, which can be used for remeshing. A good remeshing algorithm should keep sharp features of the mesh while maintaining samples uniformly distributed. Our algorithm can easily generate uniform samples due to our blue noise property. As discussed in Fig. 9.10 our algorithm will naturally place points in regions with sharp features.

Fig. 10.1 samples a bowl surface and mesh the result using Ball Pivoting algorithm (Bernardini *et al.* 1999). Our algorithm preserves the features of the bowl without the need of any additional techniques such as sub-sampling in regions of high curvature, thanks to the properties



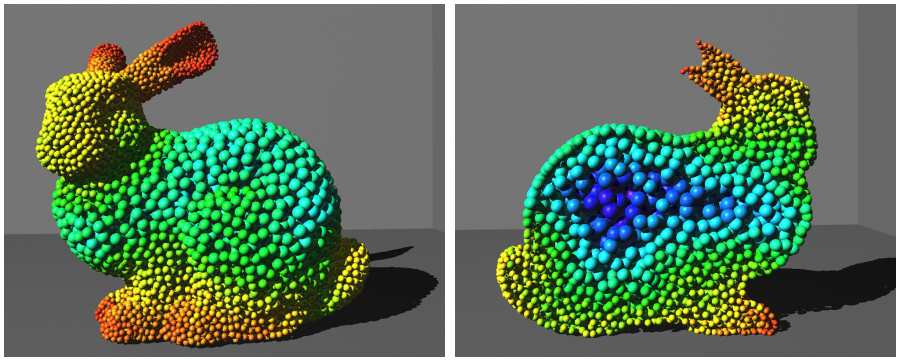
(a) Chen *et al.* (2013) (b) SPH Surface Sampling

Figure 10.1: *Remeshing results of surface sampling. The surface of the bowl is remeshed with 5000 samples. With our method, points are implicitly attracted to regions of high curvature.*

of SPH sampling described in Sec. 9.2. The surface remeshing results of Chen *et al.* (2013) are compared with ours. Our method turns out better on sharp feature preservation. Further improvement of this result with adaptive sampling based on surface curvature is a promising research direction.

10.3 Adaptive Volume Sampling

Adaptive volume sampling according to different features can be very useful for a variety of applications, such as variational tetrahedralization and volume rendering. SPH sampling allows us to choose a different size function based on a particular application. Fig. 10.2 samples the interior and surface of the bunny adaptively by determining the feature size using the medial axis of the manifold, similar to Adams *et al.* (2007). The scale function of the adaptive sampling is set according to the computed size function. Other size functions, such as that of Alliez *et al.* (2005) are also applicable.



(a) (b)

Figure 10.2: *The bunny model is adaptively sampled based on a size function deduced using the medial axis (Adams* et al. *2007), resulting in dense sampling of the ears, tail and feet, and sparse sampling of the body. The cutaway demonstrates that both the surface and internal volume is well adaptively sampled. The radius and color of particles is set according to the size function.*

10.4 Summary

The chapter mainly focuses on the applications of SPH sampling. In Chapters 4, 5 and 6, the dissolution model is introduced. In Chapters 7, 8, 9 and 10, the new sampling method based on SPH is presented which efficiently provides controllable blue noise spectral profiles. In the next chapter the use of SPH sampling with CCVT profile in the dissolution simulation will be discussed .

Chapter 11

SPH sampling with Dissolution Simulation

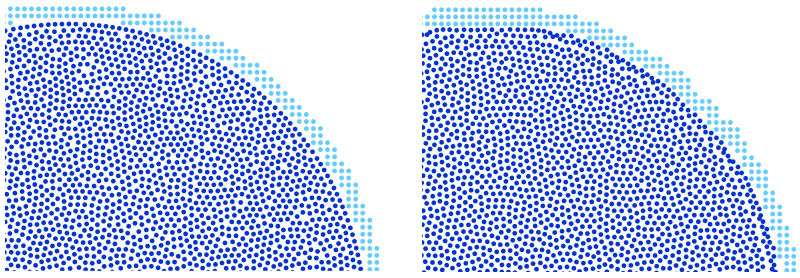
Dissolution simulation is both the motivation of our SPH sampling algorithm and one of the applications of it. During fluid-solid interaction, especially when the solid shape is changing, the representation of the solute becomes vital. The choice of the sampling methods not only affects the rendering results, but also influences the correctness of the simulation.

Compared with simple grid sampling, SPH sampling takes the advantage of anti-aliasing, which gives better results for the boundary representation and is also more friendly to use in SPH-based dissolution simulation compared with other blue noise sampling methods.

11.1 Solute Sampling

The random sampling of the solute could cause the penetration of fluid particles into solid particles thus cannot guarantee the correct boundary criterion. The excessive regularity of the simple grid sampling brings about the aliasing artefacts. Our SPH sampling with CCVT profiles generates samples in an even yet stochastic distribution which avoids the penetration and eliminates the artefacts. It shares the blue noise

properties benefiting the rendering appearance and smoothing the simulation process.



(a) (b)

Figure 11.1: *Simple grid sampling artefacts (reproduced from Schechter and Bridson (2012)). Light blue particles are the ghost particles and the dark blue ones represent the fluid.*

Schechter and Bridson (2012) pointed out the artefacts caused by using simple grid sampling for the distribution of the ghost particles, shown in the Fig. 11.1. Fig. 11.1(a) is the initial configuration, and after 500 frames in Fig. 11.1(b), it can be clearly seen that errors are evident in the fluid simulation near the interface. This phenomenon might not be obvious in our dissolution simulation since the solid solute is shrinking.

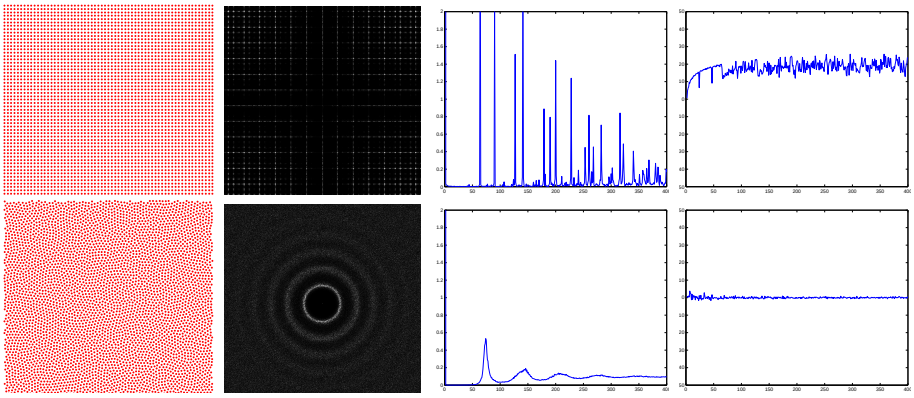


Figure 11.2: *Fourier spectrum analysis of uniform grid sampling (top row) and SPH sampling (bottom row). The images from left to right are respectively sample points, spectrum, radial mean and anisotropy.*

Introducing artefacts in fluid behaviours is not the only problem simple grid sampling may cause, but also the anisotropy problem. This can be explained using Fourier frequency transform of the grid sampling,

shown in Fig. 11.2, compared with the Fourier frequency transform of the SPH sampling with CCVT profile. Grid sampling has a considerable high oscillation which means high structured aliasing due to the excessive regularity. The anisotropy of grid sampling is also significantly high while the SPH sampling is almost nil and flat. Anisotropy is the property of directionally dependent, which implies identical properties in all directions. In other words, the SPH sampling is isotropic while simple grid sampling is not.

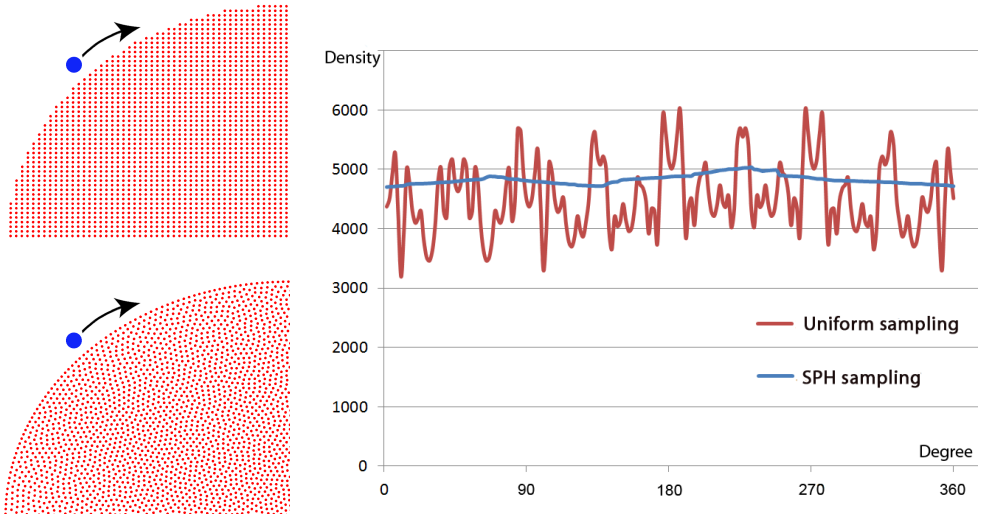


Figure 11.3: *A blue-coloured fluid particle moving along the solid boundary. The red particles are the solid solute. The right graph shows the density changes while the fluid particle is moving along the spherical solute using uniform sampling and SPH sampling.*

Imagine a fluid particle is moving along the boundary of a solute distributed using simple grid sampling or SPH sampling, shown in the left images of Fig. 11.3. The density contribution from the solute particles within the neighbourhood of the fluid particle is calculated. In the example of grid sampling, the density is oscillating heavily while the particle is rounding the spherical solute, shown as the red curve in the right image of Fig. 11.3. In contrast, while the fluid particle is rounding our SPH sampled solute, the density difference is subtle, shown as the blue curve in the right image of Fig. 11.3. The difference enlarges when the smooth radius is smaller. The error of density estimation will increase the inaccuracy of the energy calculation during dissolution, leading to

(a) Poisson disk sampling (b) Poisson disk sampling with relaxation (Schechter and Bridson 2012) (c) Our SPH sampling

Figure 11.4: *A comparison of solute sampling methods using Poisson disk sampling, Poisson disk sampling with relaxation (Schechter and Bridson 2012) and our SPH sampling sampling. All these sampling methods have the same iterations of 30. The colour represents the density of each particle. Note that irregular density along the boundary would cause surface penetrations.*

discontinuous behaviours. Therefore, isotropic sampling methods are needed which tightly fits given solute boundary. SPH sampling maintains the density stability even when the smooth length is reasonable small which potentially allows speeding up the simulation.

Alternative isotropic sampling techniques have been experimented . Compared with the Poisson sampling used in Schechter and Bridson (2012) our SPH sampling minimizes the internal pressure of the solid solute and achieves a more uniform distribution. The results are shown in Fig. 11.4. The colour coded particles demonstrate the density. Schechter and Bridson (2012) sampled both the interior and the surface using Poisson disk patterns similar to Bridson (2007). To create N samples, the time complexity is $O(2N - 1)$. They dealt with the surface and interior asynchronous, using an initial surface sampling as seeds to sample the volume, which meant at least two times of the original time complexity. However, our method solves both surface and volume sampling simultaneously, the time complexity of our method for both surface and volume sampling will be $O(N)$. Therefore, our sampling method not only achieves better results but also reduces the algorithmic complexity and cuts the computation cost.

Compared with other blue noise sampling methods, the use of SPH sampling in our SPH dissolution model also gives an extra bonus — the density control of the undissolved and dissolved solute particles. Take ice melting to water as an example, the density of the undissolved solute can be controlled by setting a corresponding sampling number to produce ice density ($0.9340g/cm$); and a correct density can be chosen for the dissolved solute, with water density ($1,000kg/m$); thus the volume change from the ice to water can be precisely mimicked by controlling the correct density ratio between the undissolved solute and dissolved solute. Fig. 11.5 displays comparison examples of dissolving bunny at the same frame with different solute densities before or after dissolving. Users can choose the specific density ratio to control the volume change of the solute before and after dissolving, thus achieving desired visual effects.

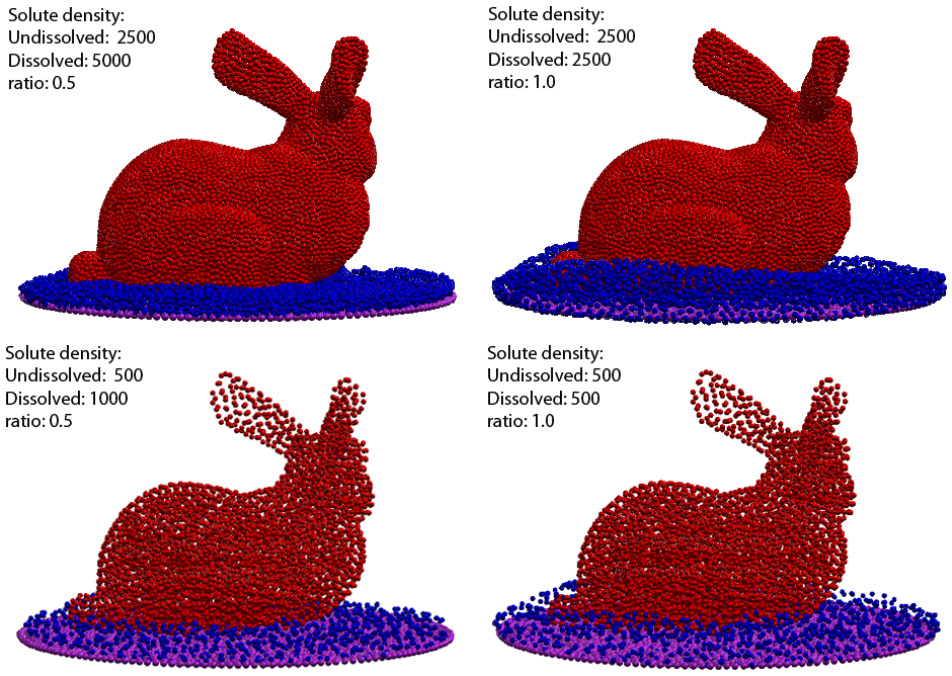


Figure 11.5: *Control of the volume change between the undissolved solute and dissolved solute using density ratio between them.*

11.2 SPH Sampling in Dissolution

Fig. 11.6 compares three different sampling methods — random, grid and SPH sampling for solute particle distribution in dissolution simulation. It is easily noticed that random sampling results a faster dissolution process because the irregular sampling cannot guarantee the boundary condition, leading more particle to collide with the boundary particles. Thanks to our sampling independent dissolution model, the simple grid sampling and SPH sampling roughly have the same dissolution pace in this experiment. However, they do present different surface conditions during the process. The grid sampling not only causes the energy estimation error, but also introduces serrated effects to the appearance of the solute surface.

In real life, the dissolution results in a rough surface of solute, shown in Fig. 11.7, because the molecular collision is a rather random process described in *Collision Theory*. Our SPH sampling mimics the roughness of the surface while the simple grid sampling causes the serrated artefacts on the surface of the tablet. Our SPH sampling certainly achieves better rendering results in high resolution. It is noteworthy that in all of these methods the sampling spacing has to be smaller than the smooth length h, otherwise the fluid-solid no-penetration criterion will fail, since the fluid could miss the density contributions from the solids.

Figure 11.7: *Tablet dissolving in reality (Anonymous 2015).*

11.3 SPH Sampling in Melting

A temperature is introduced for simulation melting in the existed dissolution framework discussed in Chapter 5. Fig. 11.9 shows a bunny melting under different temperatures. The model is simplified by only

considering the heat transfer from the circle pan to the bunny and ignoring the heat transfer elsewhere, basically, a melting model which is dissolving with heat. The energy calculation of Eq. 5.5 is transformed by replacing the kinetic energy with temperature θ as below:

$$E_s = m_{vs} \sum_h \theta W(\mathbf{x}_s - \mathbf{x}_h, h), \tag{11.1}$$

Where m_{vs} is the volume–corrected mass, $\mathbf{x}_s$ and $\mathbf{x}_h$ are respectively the position of the solute particle and the heated particle which are the particles representing the pan. When the heat energy of the solid particle exceeds a user defined threshold, the particle will detach from the solid object and become fluid particle. SPH sampling is used for our solute representation.

This example is also tested using simple grid sampling, the serrated artefacts seem more obvious in 3D examples. Fig. 11.8 shows a bunny with grid sampling melting in a heated pan. Since one layer of particles will have the same energy (no randomness), clusters of particles will gain the same heat and melt at the same time, leading to obvious popping artefacts during the melting process.

11.4 Summary

This chapter uses SPH sampling for solute particle distribution in dissolution simulation. Compared with simple grid sampling, SPH sampling takes the advantage of anti-aliasing, which achieves better visual results for the surface representation. Compared with other blue noise sampling method, SPH sampling is more efficient and can be achieved easily in high dimensions. Furthermore, it is user friendly in SPH-based dissolution simulation which allows users to control the volume change of the solute before and after dissolving.

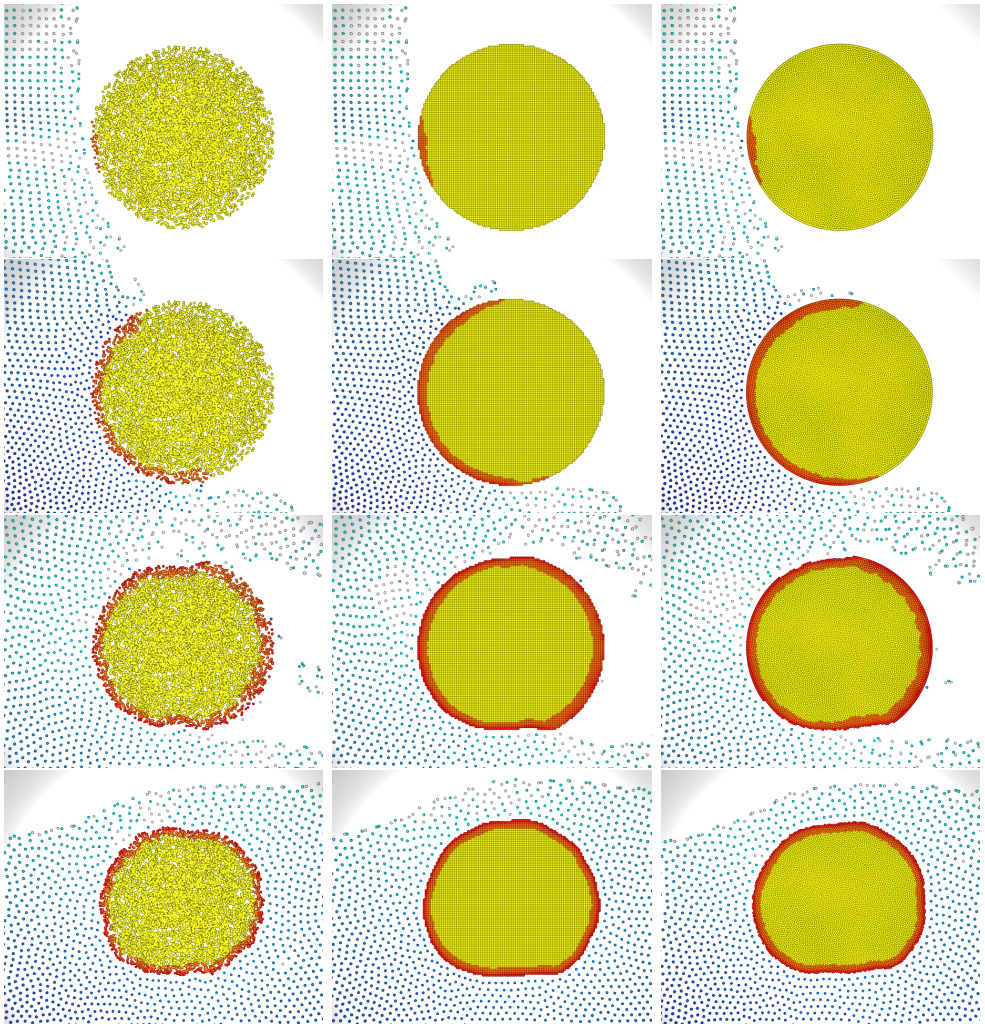


Figure 11.6: *Comparison of different sampling methods for solute in dissolution simulation at frame 100, 300, 1000 and 10000 (from top to bottom). The left images are random sampling; the middle images are simple grid sampling; and the right images are the SPH sampling with CCVT profile. The solute spheres are all sampled with 8000 particles.*

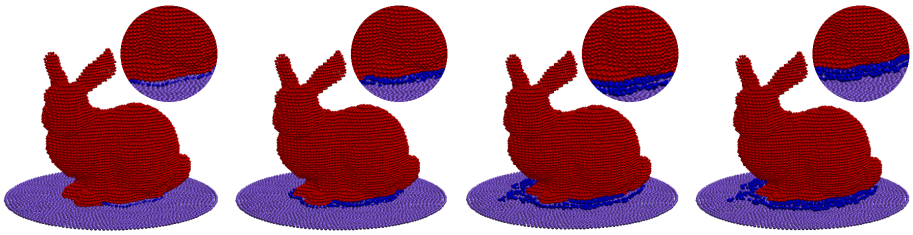


Figure 11.8: *Serrated artefacts of bunny melting. The bunny is sampled using simple grid and the particles melt layer by layer.*

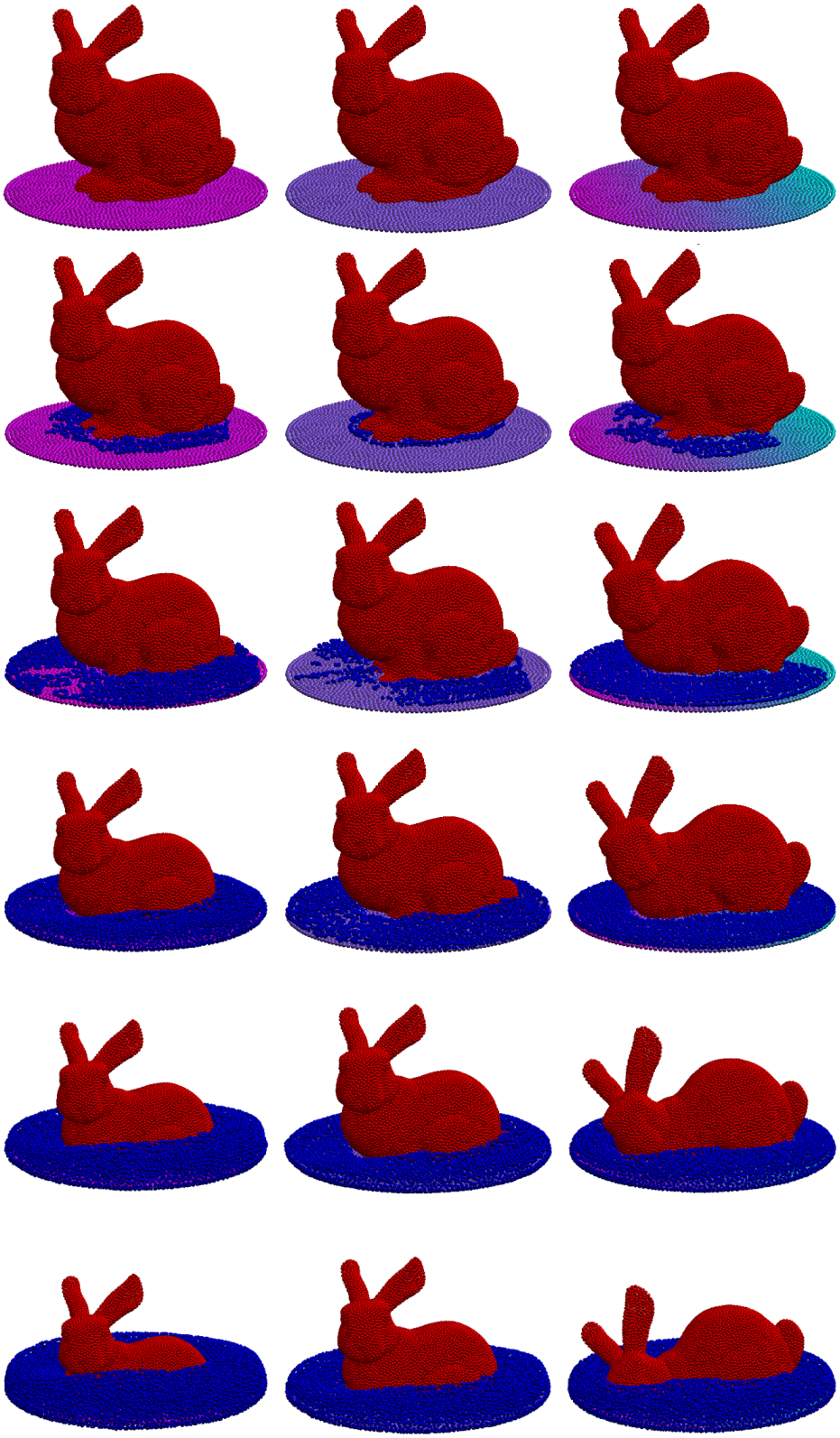


Figure 11.9: *The left column shows a bunny melting in a high temperature pan, while the middle one melts in a low temperature pan, and the right one melts in a pan whose temperature is various from one side to the other (purple particles have high temperature and green ones have low temperature).*

Chapter 12

Conclusion and Future Work

In this thesis, an energy-based method is proposed for dissolution simulation using Smoothed Particle Hydrodynamic Sampling for both the solid solute and fluid solvent.

Firstly, a particle based framework is introduced for complex fluid–solid interaction — dissolution simulation which is derived from chemical Collision Theory which is used as an analogy to the dissolution process modelling. The key to this model is a new expression for particle excitation based on kinetic energy. By introducing the volume–corrected mass instead of mass to update local particle energy, it is able to ensure the consistent dissolution processes even when the sampling resolutions of the solute are different. The relationship between global dissolution time and the activation energy is also identified, allowing animators to easily adjust the global properties of the dissolution simulation by modifying just one single parameter. Our approach ensures that dissolution behaviour is physically and chemically plausible. In addition, our framework is simple to implement and is straightforward to fit in an existing particle–based fluid framework, and naturally lends itself to a parallel implementation.

Secondly, a new blue noise sampling method — SPH sampling which improves the solute particle distribution is proposed. The particle–based representation for the solute naturally supports objects of arbitrary

topology and the phenomenon of object separation during dissolution. Compared with random distribution, our SPH sampling guarantees the correct boundary criterion by preventing fluid particles from penetrating the solute. Compared with a simple grid distribution, our method avoids the aliasing artefacts and gives smoother rendering results. Compared with other blue noise sampling methods, the conjunction of our SPH sampling method with SPH dissolution framework allows user defined volume change of the solute before and after dissolving.

This novel sampling method also addresses the limitations of classic blue noise methods, such as heavy computational cost, difficulty in extending to general sampling domains such as surfaces and volumes, and controllability of the distribution properties of the generated samples. Our SPH sampling method achieves fast sampling in general dimensions. By varying a single parameter our method can generate a variety of blue noise samples with different distribution properties, ranging from Lloyds relaxation to Capacity Constrained Voronoi Tessellations (CCVT), suitable for applications such as image stippling and re-meshing.

All the work and our findings to date naturally lead to the following areas for future work:

12.1 Improvement of Dissolution Model

Diffuse material such as foam, spray and bubbles are involved to add the reality of fluid simulation. When some chemical reaction happens, it may be associated with bubbles, which will rise up, stack upon the surface of the fluid and become foam. In our thesis, bubble is simulated with only simple Brownian motion. More realistic foam simulation is intended to enhance the reality of dissolution model, such as using Voronoi diagram to simulate foams and bubbles (Busaryev *et al.* 2012b).

The unified particle nature of our dissolution model and other recently published work hint at a unified representation for interacting media, allowing gas, solids and fluids to be simulated within a single framework.

This approach makes possible to extend to other natural phenomenon such as evaporation and boiling. Melting is simulated in a very simple model based on our dissolution framework. More sophisticated melting simulation can be achieved by considering the complex heat transfer between particles, such as Lii and Wong (2014).

The adaptive methods for fluid simulation, such as Adams *et al.* (2007) and Solenthaler and Gross (2011) improve the simulation speed while still preserving the visual quality. From this concern, it is our intention to build up an adaptive dissolution simulation framework in which fluid particles are densely sampled at the dissolution site while sparsely sampled otherwise.

12.2 Further Investigation of SPH Sampling

There are several exciting avenues for research arising from our SPH sampling. An interesting problem is how the sampling patterns are affected by the choice of kernels. Tab. 12.1 shows experimental results of different kernels and their corresponding points distributions. In this experiment the kernel gradient of the pressure force (which controls the mapping from the density difference to the force) is replaced with four typical kernels: a box kernel, tent kernel, quadratic kernel and and off–center double-peak kernel. The results show that sampling patterns are directly influenced by the kernel choice. The tent kernel and quadratic kernel give results similar with blue noise distributions, but the other kernels generate distributions that are more like pink or green noise.

Further experiments and theoretical analysis would be needed to understand the relationship between the kernel and the resulting distribution. An interesting problem is to study how to automatically compute, given a target spectral distribution function, a kernel function that would lead to that target distribution. This would make it possible to use our SPH-based method to generate samples with an arbitrarily designed spectrum by the user. Therefore, the relationship between the kernel and its resulting sample spectrum warrants further investigation.

Another possible direction is to study how to improve the trade-off between Nyquist Frequency and oscillation. The trade-off of our algorithm is currently not competitive with those of ideal blue noise or step blue noise and the single-peak blue noise discussed in Heck *et al.* (2013).

Finally, it is our intention to try other fluid simulation frameworks for the blue noise sampling and to study their trade-off tendency and potential performance improvement.

12.3 Medical Applications of Current Model

Fluid simulations can also be used in medical applications. There are few research papers discussed the problem of blood simulation with a fluid simulation scheme. Müller *et al.* (2004) simulated the vessel injury caused by pressure forces. However, they used triangles to represent the solid boundaries which restricted the local deformation, resulting in artefacts. Dzwinel *et al.* (2003) described a discrete-particle model for the flow of plasma and Red Blood Cells (RBC) with computational fluid dynamics. The flexible viscoelastic RBC and the walls of the elastic vessel were made up of solid particles held together by elastic harmonic forces. In Walsh *et al.* (2005), the blood was assumed to be incompressible, homogeneous and Newtonian fluid.

Our dissolution model has a potential connection with the medical problem of blood clotting. The formation of blood clots in a vein is extremely dangerous threatening people's life. Our focus will be on the simulation of how to efficiently dissolve the blood clots, which is an extension of our current dissolution model. To the best of our knowledge there is hardly any computer graphics paper relating to the blood clots. It can be potentially used in pharmacy test and become a significant application of the dissolution model and is a topic which warrants further scrutiny.

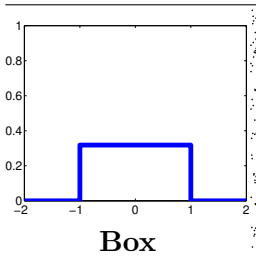
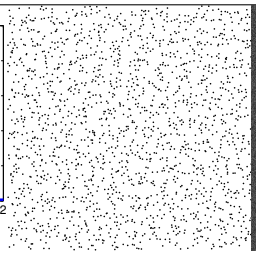
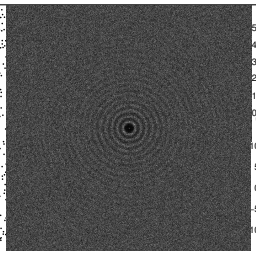
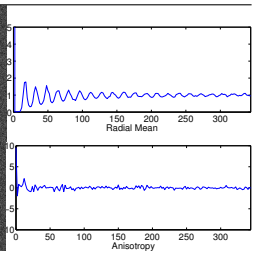
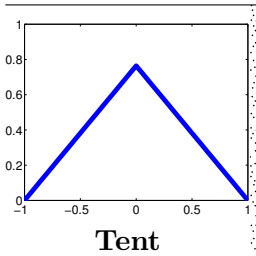
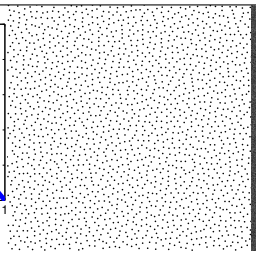
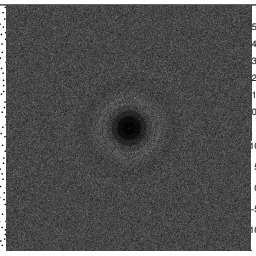
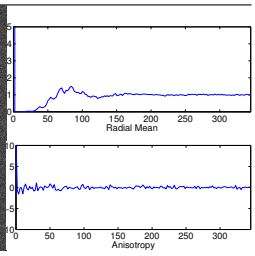
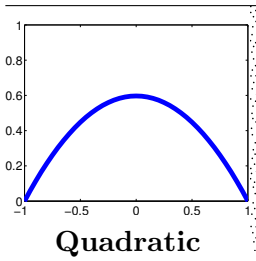
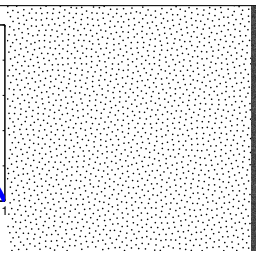
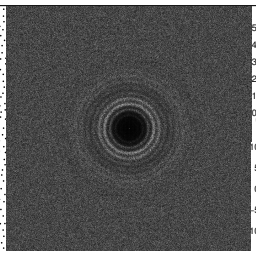
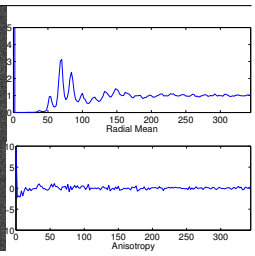
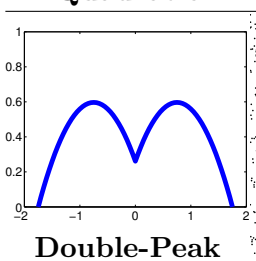
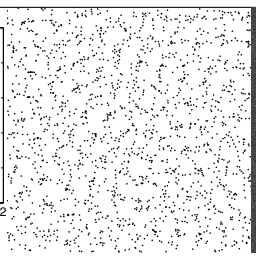
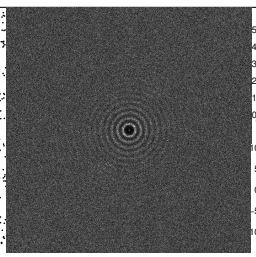
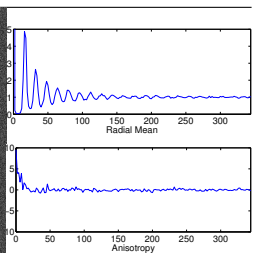
Kernel	Samples	Spectrum	Radial Mean & Anisotropy
 Box			
 Tent			
 Quadratic			
 Double-Peak			

Table 12.1: *Experimental studies of how different kernels affects the distribution of the resulting samples. Here four kernels with different shapes, the corresponding samples, and Fourier spectrum analysis results are shown.*

Appendix A

List of publications

Jiang, M., Southern, R., Tharib, S. and J, Zhang., 2013. Energy-Based Dissolution Simulation. In *2013 International Conference on Computer-Aided Design and Computer Graphics*,CAD/Graphics 2013, Guangzhou, China, pp 405-406.

Jiang, M., Southern, R. and J, Zhang., 2015. A Particle-based Dissolution Model using Chemical Collision Energy. In *Proceedings of the 10th International Conference on Computer Graphics Theory and Applications*, GRAPP 2015, Berlin, Germany, pp 285-293.

Jiang, M., Zhou, Y., Wang, R., Southern, R. and J, Zhang., 2015. Blue noise sampling using an SPH-based method. *ACM Trans. Graph.* 34(6), pp 211:1-211:11.

Bibliography

Adami S., Hu X. and Adams N. A., 2013. A transport-velocity formulation for smoothed particle hydrodynamics. *Journal of Computational Physics*, **241**, 292–307.

Adams B., Pauly M., Keiser R. and Guibas L. J., 2007. Adaptively sampled particle fluids. In *ACM Transactions on Graphics (TOG)*, volume 26. ACM, 48.

Akinci N., Akinci G. and Teschner M., November 2013. Versatile surface tension and adhesion for SPH fluids. *ACM Trans. Graph.*, **32**(6), 182:1–182:8.

Akinci N., Ihmsen M., Akinci G., Solenthaler B. and Teschner M., July 2012. Versatile rigid-fluid coupling for incompressible SPH. *ACM Trans. Graph.*, **31**(4), 62:1–62:8.

Alliez P., Cohen-Steiner D., Yvinec M. and Desbrun M., July 2005. Variational tetrahedral meshing. *ACM Trans. Graph.*, **24**(3), 617–625.

Amada T. 2006. 189–205. Real-time particle-based fluid simulation with rigid body interaction. volume 6.

Anderson J. D. and Wendt J., 1995. *Computational fluid dynamics*, volume 206. Springer.

Anonymous , 2015. Science fair projects: Dissolving tablets more quickly. `http://www.all-science-fair-projects.com/print_project_1302_134` Accessed: 2016-01-19.

Aristodemo F., Marrone S. and Federico I., 2015. SPH modeling of plane

jets into water bodies through an inflow/outflow algorithm. *Ocean Engineering*, **105**, 160–175.

Auer S., 2008. Realtime particle-based fluid simulation. *Computer Graphics and Visualization*.

Baatz M. and Schäpe A., 2000. Multiresolution segmentation: an optimization approach for high quality multi-scale image segmentation. *Angewandte Geographische Informationsverarbeitung XII*, **58**, 12–23.

Bærentzen J. A. and Aanæs H., 2002. Generating signed distance fields from triangle meshes. *Informatics and Mathematical Modeling, Technical University of Denmark, DTU*, **20**.

Baerentzen J. A. and Aanaes H., May 2005. Signed distance computation using the angle weighted pseudonormal. *IEEE Transactions on Visualization and Computer Graphics*, **11**(3), 243–253.

Balzer M., Schlömer T. and Deussen O., July 2009. Capacity-constrained point distributions: A variant of Lloyd's method. *ACM Trans. Graph.*, **28**(3), 86:1–86:8.

Batty C., Bertails F. and Bridson R., July 2007. A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph.*, **26** (3).

Becker M. and Teschner M., 2007. Weakly compressible SPH for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, 209–217.

Becker M., Tessendorf H. and Teschner M., May 2009. Direct forcing for lagrangian rigid-fluid coupling. *IEEE Transactions on Visualization and Computer Graphics*, **15**(3), 493–503.

Ben-Ezra M., Lin Z. and Wilburn B., 2007. Penrose pixels super-resolution in the detector layout domain. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE, 1–8.

Beneš B., Těšínský V., Hornyš J. and Bhatia S., 2006. Hydraulic erosion. *Computer Animation and Virtual Worlds*, **17**(2), 99 – 108.

Bernardini F., Mittleman J., Rushmeier H., Silva C. and Taubin G., October 1999. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, **5**(4), 349–359.

Blelloch G. E., November 1989. Scans as primitive parallel operations. *IEEE Trans. Comput.*, **38**(11), 1526–1538.

Bodin K., Lacoursié C. and Servin M., 2012. Constraint fluids. *Visualization and Computer Graphics, IEEE Transactions on*, **18**(3), 516–526.

Bowers J., Wang R., Wei L.-Y. and Maletz D., December 2010. Parallel Poisson disk sampling with spectrum analysis on surfaces. *ACM Trans. Graph.*, **29**(6), 166:1–166:10.

Bracewell R., 1965. The Fourier transform and its applications. *New York*, **5**.

Brackbill J. U. and Ruppel H. M., August 1986. FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *J. Comput. Phys.*, **65**(2), 314–343.

Bridson R., 2007. Fast Poisson disk sampling in arbitrary dimensions. In *ACM SIGGRAPH 2007 sketches*, SIGGRAPH '07, New York, NY, USA. ACM.

Bridson R., 2008. *Fluid simulation for computer graphics*. CRC Press.

Bridson R. and Müller-Fischer M., 2007. Fluid simulation: Siggraph 2007 course notesvideo files associated with this course are available from the citation page. In *ACM SIGGRAPH 2007 Courses*, SIGGRAPH '07, New York, NY, USA. ACM, 1–81.

Busaryev O., Dey T. K., Wang H. and Ren Z., July 2012a. Animating bubble interactions in a liquid foam. *ACM Trans. Graph.*, **31**(4), 63:1–63:8.

Busaryev O., Dey T. K., Wang H. and Ren Z., July 2012b. Animating bubble interactions in a liquid foam. *ACM Trans. Graph.*, **31**(4), 63:1–63:8.

Carlson M., Mucha P. J. and Turk G., 2004. Rigid fluid: animating the interplay between rigid bodies and fluid. volume 23. ACM, 377–384.

CDER , 2004. Guidance for industry: Dissolution testing of immediate release solid oral dosage forms. `http://www.fda.gov/downloads/drugs/guidancecomplianceregulatoryinformation/guidances/ucm070237.pdf` Accessed: 2016-01-19.

Chen J., Ge X., Wei L.-Y., Wang B., Wang Y., Wang H., Fei Y., Qian K.-L., Yong J.-H. and Wang W., November 2013. Bilateral blue noise sampling. *ACM Trans. Graph.*, **32**(6), 216:1–216:11.

Chen Q. and Xu W., 1998. A zero-equation turbulence model for indoor airflow simulation. *Energy and buildings*, **28**(2), 137–144.

Chen X. B., Lee H. P., Chong H., Fook V. and Wang D. Y., 2009. Assessment of septal deviation effects on nasal air flow: a computational fluid dynamics model. *The Laryngoscope*, **119**(9), 1730–1736.

Chen Z., Yuan Z., Choi Y.-K., Liu L. and Wang W., October 2012. Variational blue noise sampling. *IEEE Transactions on Visualization and Computer Graphics*, **18**(10), 1784–1796.

Chorin A. J., 1968. Numerical solution of the Navier-Stokes equations. *Mathematics of computation*, **22**(104), 745–762.

Chorin A. J., 1973. Numerical study of slightly viscous flow. *Journal of fluid mechanics*, **57**(04), 785–796.

Clark J., 2004. *The Essential Dictionary of Science*. New York: Barnes & Noble Book.

Clay Mathematics Institute , 2016. Millennium problems: Navier-Stokes equation. `http://www.claymath.org/millennium-problems` Accessed: 2016-02-17.

Cleary P. W., Pyo S. H., Prakash M. and Koo B. K., 2007. Bubbling and frothing liquids. *ACM Transactions on Graphics (TOG)*, **26**(3), 97.

Cohen M. F., Shade J., Hiller S. and Deussen O., July 2003. Wang

tiles for image and texture generation. *ACM Trans. Graph.*, **22**(3), 287–294.

Condit R., Ashton P. S., Baker P., Bunyavejchewin S., Gunatilleke S., Gunatilleke N., Hubbell S. P., Foster R. B., Itoh A., LaFrankie J. V., Lee H. S., Losos E., Manokaran N., Sukumar R. and Yamakura T., 2000. Spatial patterns in the distribution of tropical tree species. *Science*, **288**(5470), 1414–1418.

Cook R. L., January 1986. Stochastic sampling in computer graphics. *ACM Trans. Graph.*, **5**(1), 51–72.

Cornelis J., Ihmsen M., Peer A. and Teschner M., May 2014. IISPH-FLIP for incompressible fluids. *Comput. Graph. Forum*, **33**(2), 255–262.

Crespo A. C., Dominguez J. M., Barreiro A., Gómez-Gesteira M. and Rogers B. D., 2011. GPUs, a new tool of acceleration in CFD: efficiency and reliability on smoothed particle hydrodynamics methods. *PLoS One*, **6**(6), e20685.

Crespo A. J., Domínguez J. M., Rogers B. D., Gómez-Gesteira M., Longshaw S., Canelas R., Vacondio R., Barreiro A. and García-Feal O., 2015. DualSPHysics: Open-source parallel CFD solver based on smoothed particle hydrodynamics (SPH). *Computer Physics Communications*, **187**, 204–216.

Cummins S. J. and Rudman M., 1999. An SPH projection method. *Journal of computational physics*, **152**(2), 584–607.

Cunningham L. S., Rogers B. D. and Pringgana G., 2014. Tsunami wave and structure interaction: an investigation with smoothed-particle hydrodynamics. *Proceedings of the Institution of Civil Engineers - Engineering and Computational Mechanics*, **167**(3), 126–138.

Dale M. R. T., Dixon P., Fortin M.-J., Legendre P., Myers D. E. and Rosenberg M. S., October 2002. Conceptual and mathematical relationships among methods for spatial analysis. *Ecography*, **25**(5), 558–577.

de Goes F., Breeden K., Ostromoukhov V. and Desbrun M., November

2012. Blue noise through optimal transport. *ACM Trans. Graph.*, **31** (6), 171:1–171:11.

de Goes F., Wallez C., Huang J., Pavlov D. and Desbrun M., July 2015. Power particles: An incompressible fluid solver based on power diagrams. *ACM Trans. Graph.*, **34**(4), 50:1–50:11.

Desbrun M. and Gascuel M.-P., 1996. Smoothed particles: A new paradigm for animating highly deformable bodies. In *Proceedings of the Eurographics Workshop on Computer Animation and Simulation '96*, New York, NY, USA. Springer-Verlag New York, Inc., 61–76.

Dippé M. A. Z. and Wold E. H., July 1985. Antialiasing through stochastic sampling. *SIGGRAPH Comput. Graph.*, **19**(3), 69–78.

Du Q. and Emelianenko M., 2006. Acceleration schemes for computing centroidal voronoi tessellations. *Numerical Linear Algebra with Applications*, **13**(2-3), 173–192.

Dunbar D. and Humphreys G., 2006. A spatial data structure for fast Poisson-disk sample generation. In *ACM Transactions on Graphics (TOG)*, volume 25. ACM, 503–508.

Dzwinel W., Boryczko K., Yuen D. A. and others , 2003. A discrete-particle model of blood dynamics in capillary vessels. *Journal of colloid and interface science*, **258**(1), 163–173.

Elcott S., Tong Y., Kanso E., Schröder P. and Desbrun M., 2007. Stable, circulation-preserving, simplicial fluids. *ACM Transactions on Graphics (TOG)*, **26**(1), 4.

Eyi S., Hager J. and Lee K., 1994. Airfoil design optimization using the Navier-Stokes equations. *Journal of Optimization Theory and Applications*, **83**(3), 447–461.

Fattal R., July 2011. Blue-noise point sampling using kernel density model. *ACM Trans. Graph.*, **30**(4), 48:1–48:12.

Featherstone R., 2014. *Rigid body dynamics algorithms*. Springer.

Fedkiw R., Stam J. and Jensen H. W., 2001. Visual simulation of smoke.

In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, New York, NY, USA. ACM, 15–22.

Fernando R., 2004. *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*. Pearson Higher Education.

Foster N. and Metaxas D., 1996. Realistic animation of liquids. *Graphical models and image processing*, **58**(5), 471–483.

Fournier A. and Reeves W. T., August 1986. A simple model of ocean waves. *SIGGRAPH Comput. Graph.*, **20**(4), 75–84.

Fourtakas G. and Rogers B., 2016. Modelling multi-phase liquid-sediment scour induced by rapid flows using smoothed particle hydrodynamics (SPH) accelerated with a graphics processing unit (GPU). *accepted for publication*.

Génevaux O., Habibi A. and Dischler J.-M., 2003. Simulating fluid-solid interaction. In *Graphics Interface*, volume 2003. Citeseer, 31–38.

Gingold R. A. and Monaghan J. J., 1977. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly notices of the royal astronomical society*, **181**(3), 375–389.

Green S., 2010. Particle simulation using CUDA. `http://www.dps.uibk.ac.at/~cosenza/teaching/gpu/nv_particles.pdf` Accessed: 2016-03-10.

Greene D. F. and Johnson E. A., 1989. A model of wind dispersal of winged or plumed seeds. *Ecology*, **70**(2), 339–347.

Guo J., Yan D.-M., Jia X. and Zhang X., 2015. Efficient maximal Poisson-disk sampling and remeshing on surfaces. *Computers and Graphics*, **46**(0), 72 – 79. Shape Modeling International 2014.

Happ P. N., Feitosa R. Q., Bentes C. and Farias R., Nov 2013. A region-growing segmentation algorithm for GPUs. *IEEE Geoscience and Remote Sensing Letters*, **10**(6), 1612–1616.

Harada T., Koshizuka S. and Kawaguchi Y., 2007a. Smoothed particle

hydrodynamics on GPUs. In *Computer Graphics International*. SBC Petropolis, 63–70.

Harada T., Tanaka M., Koshizuka S. and Kawaguchi Y., 2007b. Real-time coupling of fluids and rigid bodies. *Proc. of the APCOM*.

Harlow F. H. and Welch J. E., 1965. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids*, **8**(12), 2182–2189.

Harris M., Sengupta S. and Owens J. D., 2007. Parallel prefix sum with CUDA. `http://http.developer.nvidia.com/GPUGems3/gpugems3_ch39.html` Accessed: 2016-02-17.

He X., Liu N., Li S., Wang H. and Wang G., 2012. Local Poisson SPH for viscous incompressible fluids. In *Computer Graphics Forum*, volume 31. Wiley Online Library, 1948–1958.

Heck D., Schlömer T. and Deussen O., July 2013. Blue noise sampling with controlled aliasing. *ACM Trans. Graph.*, **32**(3), 25:1–25:12.

Hegeman K., Carr N. and Miller G. 2006. 228–235. Particle-based fluid simulation on the GPU. In Alexandrov V., van Albada G., Sloot P. and Dongarra J., editors, *Computational Science ICCS 2006*, volume 3994 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg.

Hérault A., Bilotta G. and Dalrymple R., 2010. SPH on GPU with CUDA. *Journal of Hydraulic Research*, **48**(1), 74–79.

Hernquist L., 1993. Some cautionary remarks about smoothed particle hydrodynamics. *The Astrophysical Journal*, **404**, 717–722.

Hoetzlein R. C., 2014. Fast fixed-radius nearest neighbors: Interactive million-particle fluids. *GPU Technology Conference (GTC)*.

Hojjatoleslami S. and Kittler J., 1998. Region growing: a new approach. *IEEE Transactions on Image processing*, **7**(7), 1079–1084.

Hong J.-M. and Kim C.-H., July 2005. Discontinuous fluids. *ACM Trans. Graph.*, **24**(3), 915–920.

Hontanon E., Escudero M., Bautista C., Garcıa-Ybarra P. and Daza L.,

2000. Optimisation of flow-field in polymer electrolyte membrane fuel cells using computational fluid dynamics techniques. *Journal of Power Sources*, **86**(1), 363–368.

Hu X. Y. and Adams N. A., 2006. A multi-phase sph method for macroscopic and mesoscopic flows. *Journal of Computational Physics*, **213** (2), 844–861.

Huang C., Zhu J., Sun H. and Wu E., 2013. Efficient fluids simulation and rendering on GPU. In *Proceedings of the 12th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, VRCAI '13, New York, NY, USA. ACM, 25–30.

Hut P. and Makino J., 2004. Two ways to write the leapfrog. `http://www.artcompsci.org/vol_1/v1_web/node34.html` Accessed: 2015-10-07.

Ihmsen M. *Particle-based simulation of large bodies of Water with bubbles, spray and foam.* PhD thesis, Universitätsbibliothek Freiburg, 2013.

Ihmsen M., Akinci N., Akinci G. and Teschner M., jun 2012. Unified spray, foam and air bubbles for particle-based fluids. *Vis. Comput.*, **28**(6-8), 669–677.

Ihmsen M., Akinci N., Becker M. and Teschner M., 2011. A parallel SPH implementation on multi-core CPUs. *Computer Graphics Forum*, **30** (1), 99–112.

Ihmsen M., Cornelis J., Solenthaler B., Horvath C. and Teschner M., March 2014a. Implicit incompressible SPH. *IEEE Transactions on Visualization and Computer Graphics*, **20**(3), 426–435.

Ihmsen M., Orthmann J., Solenthaler B., Kolb A. and Teschner M., 2014b. SPH fluids in computer graphics. In Lefebvre S. and Spagnuolo M., editors, *Eurographics 2014 - State of the Art Reports*. The Eurographics Association.

Jensen M. H., Johansen A. and Simonsen I., 2003. Inverse fractal

statistics in turbulence and finance. *International Journal of Modern Physics B*, **17**(22n24), 4003–4012.

Jerri A. J., 1977. The shannon sampling theoremits various extensions and applications: A tutorial review. *Proceedings of the IEEE*, **65**(11), 1565–1596.

Jiang M., Southern R., Tharib S. and Zhang J., 2013. Energy-based dissolution simulation. In *2013 International Conference on Computer-Aided Design and Computer Graphics, CAD/Graphics 2013, Guangzhou, China, November 16-18, 2013*, 405–406.

Jiang M., Southern R. and Zhang J., 2015a. A particle-based dissolution model using chemical collision energy. In *GRAPP 2015 - Proceedings of the 10th International Conference on Computer Graphics Theory and Applications, Berlin, Germany, 11-14 March, 2015.*, 285–293.

Jiang M., Zhou Y., Wang R., Southern R. and Zhang J. J., 2015b. Blue noise sampling using an SPH-based method. *ACM Transactions on Graphics (TOG)*, **34**(6), 211.

Jones M. W., Bærentzen J. A. and Sramek M., 2006. 3D distance fields: A survey of techniques and applications. *Visualization and Computer Graphics, IEEE Transactions on*, **12**(4), 581–599.

Josephson B. D., 1995. *A trans-human source of music?* Finnish Artificial Intelligence Society, Helsinki.

Kalantari N. K. and Sen P., 2011. Efficient computation of blue noise point sets through importance sampling. In *Proceedings of the Twenty-second Eurographics Conference on Rendering*, EGSR '11, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association, 1215–1221.

Kalojanov J. and Slusallek P., 2009. A parallel algorithm for construction of uniform grids. In *Proceedings of the Conference on High Performance Graphics 2009*, HPG '09, New York, NY, USA. ACM, 23–28.

Keiser R., Adams B., Gasser D., Bazzi P., Dutré P. and Gross M., 2005. A unified lagrangian approach to solid-fluid animation. In *Proceedings of the Second Eurographics / IEEE VGTC Conference on Point-*

Based Graphics, SPBG'05, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association, 125–133.

Kelager M., 2006. Lagrangian fluid dynamics using smoothed particle hydrodynamics. `http://image.diku.dk/projects/media/kelager.06.pdf` Accessed: 2013-05-22.

Kim D., Song O.-y. and Ko H.-S., 2010. A practical simulation of dispersed bubble flow. In *ACM Transactions on Graphics (TOG)*, volume 29. ACM, 70.

Kopf J., Cohen-Or D., Deussen O. and Lischinski D., 2006. Recursive wang tiles for real-time blue noise. volume 25. ACM.

Koumoutsakos P., 1997. Inviscid axisymmetrization of an elliptical vortex. *Journal of Computational Physics*, **138**(2), 821–857.

Koumoutsakos P., Cottet G.-H. and Rossinelli D., 2008. Flow simulations using particles: Bridging computer graphics and CFD. In *ACM SIGGRAPH 2008 Classes*, SIGGRAPH '08, New York, NY, USA. ACM, 25:1–25:73.

Krajnovic S. and Davidson L., 2004. Large-eddy simulation of the flow around simplified car model. *SAE paper*, 01–0227.

Kravtchenko T., Renoir J., Parker A. and Brigand G., 1999. A novel method for determining the dissolution kinetics of hydrocolloid powders. *Food Hydrocolloids*, **13**(3), 219 – 225.

Krištof P., Beneš B., Křivánek J. and Šťava O., 2009. Hydraulic erosion using smoothed particle hydrodynamics. In *Computer Graphics Forum*, volume 28, 219–228.

Krog O. E. and Elster A. C., 2012. Fast GPU-based fluid simulations using SPH. In *Proceedings of the 10th International Conference on Applied Parallel and Scientific Computing - Volume 2*, PARA'10, Berlin, Heidelberg. Springer-Verlag, 98–109.

Kroll N., Rossow C., Schwamborn D., Becker K. and Heller G., 2002. Megaflow-a numerical flow simulation tool for transport aircraft design. In *ICAS Congress*, 1–105.

Lagae A. and Dutré P., October 2006. An alternative for wang tiles: Colored edges versus colored corners. *ACM Trans. Graph.*, **25**(4), 1442–1459.

Lagae A. and Dutré P., June 2008. Compact, fast and robust grids for ray tracing. *Computer Graphics Forum (Proceedings of the 19th Eurographics Symposium on Rendering)*, **27**(4), 1235–1244.

Lagae A. and Dutré P., 2008. A comparison of methods for generating Poisson disk distributions. In *Computer Graphics Forum*, volume 27. Wiley Online Library, 114–129.

Lau D. L., Arce G. R. and Gallagher N. C., 1998. Green-noise digital halftoning. *Proceedings of the IEEE*, **86**(12), 2424–2444.

Lee H.-Y., Hong J.-M. and Kim C.-H., 2009. Interchangeable SPH and level set method in multiphase fluids. *The Visual Computer*, **25**(5-7), 713–718.

Leimkuhler B. J., Reich S. and Skeel R. D. 1996. 161–185. Integration methods for molecular dynamics. In *Mathematical Approaches to biomolecular structure and dynamics*, Springer.

Leonard A., 1980. Vortex methods for flow simulation. *Journal of Computational Physics*, **37**(3), 289–335.

Li H., Nehab D., Wei L.-Y., Sander P. V. and Fu C.-W., 2010a. Fast capacity constrained Voronoi tessellation. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '10, New York, NY, USA. ACM, 13:1–13:1.

Li H., Wei L.-Y., Sander P. V. and Fu C.-W., December 2010b. Anisotropic blue noise sampling. *ACM Trans. Graph.*, **29**(6), 167:1–167:12.

Lii S.-Y. and Wong S.-K., 2014. Ice melting simulation with water flow handling. *The Visual Computer*, **30**(5), 531–538.

Lind S., Xu R., Stansby P. and Rogers B., 2012. Incompressible smoothed particle hydrodynamics for free-surface flows: a generalised diffusion-based algorithm for stability and validations for impulsive flows and

propagating waves. *Journal of Computational Physics*, **231**(4), 1499–1523.

Lloyd S., September 1982. Least squares quantization in PCM. *IEEE Trans. Inf. Theor.*, **28**(2), 129–137.

Longshaw S. and Rogers B., 2015. Automotive fuel cell sloshing under temporally and spatially varying high acceleration using GPU-based smoothed particle hydrodynamics (SPH). *Advances in Engineering Software*, **83**, 31 – 44.

Losasso F., Shinar T., Selle A. and Fedkiw R., July 2006. Multiple interacting liquids. *ACM Trans. Graph.*, **25**(3), 812–819.

Losasso F., Talton J., Kwatra N. and Fedkiw R., July 2008. Two-way coupled SPH and particle level set fluid simulation. *IEEE Transactions on Visualization and Computer Graphics*, **14**(4), 797–804.

Lucy L. B., December 1977. A numerical approach to the testing of the fission hypothesis. *Astron. J.*, **82**, 1013–1024.

Macia F., González L. M., Cercos-Pita J. L. and Souto-Iglesias A., 2012. A boundary integral sph formulation consistency and applications to isph and wcsph. *Progress of Theoretical Physics*, **128**(3), 439–462.

Macklin M. and Müller M., July 2013. Position based fluids. *ACM Trans. Graph.*, **32**(4), 104:1–104:12.

Macklin M., Müller M., Chentanez N. and Kim T.-Y., 2014. Unified particle physics for real-time applications. *ACM Transactions on Graphics (TOG)*, **33**(4), 153.

Manenti S., Sibilla S., Gallati M., Agate G. and Guandalini R., 2011. Sph simulation of sediment flushing induced by a rapid water flow. *Journal of Hydraulic Engineering*, **138**(3), 272–284.

Marrone S., Colagrossi A., Antuono M., Colicchio G. and Graziani G., 2013. An accurate sph modeling of viscous flows around bodies at low and moderate reynolds numbers. *Journal of Computational Physics*, **245**, 456–475.

Marshall J., Adcroft A., Hill C., Perelman L. and Heisey C., 1997. A finite-volume, incompressible Navier Stokes model for studies of the ocean on parallel computers. *Journal of Geophysical Research: Oceans*, **102**(C3), 5753–5766.

Martinez V., Paredes S., Borgani S. and Coles P., 1995. Multiscaling properties of large-scale structure in the universe. *Science*, **269**(5228), 1245.

McCool M. and Fiume E., 1992. Hierarchical Poisson disk sampling distributions. In *Proceedings of the Conference on Graphics Interface '92*, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc., 94–105.

McNaught A. D. and Wilkinson A., 1997. *Collision theory*. Oxford: Blackwell Scientific Publications.

Mercier O., Beauchemin C., Thuerey N., Kim T. and Nowrouzezahrai D., October 2015. Surface turbulence for particle-based liquid simulations. *ACM Trans. Graph.*, **34**(6), 202:1–202:10.

Mitchell D. P., August 1987. Generating antialiased images at low sampling densities. *SIGGRAPH Comput. Graph.*, **21**(4), 65–72.

Mitchell D. P., July 1991. Spectrally optimal sampling for distribution ray tracing. *SIGGRAPH Comput. Graph.*, **25**(4), 157–164.

Mokos A., Rogers B. D., Stansby P. K. and Domnguez J. M., 2015. Multi-phase SPH modelling of violent hydrodynamics on GPUs. *Computer Physics Communications*, **196**, 304 – 316.

Monaghan J. J., February 1994. Simulating free surface flows with SPH. *J. Comput. Phys.*, **110**(2), 399–406.

Monaghan J. J., 2005. Smoothed particle hydrodynamics. *Reports on Progress in Physics*, **68**(8), 1703.

Monaghan J., 1992. Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics*, **30**, 543–74.

Mullen P., Crane K., Pavlov D., Tong Y. and Desbrun M., July

2009. Energy-preserving integrators for fluid animation. *ACM Trans. Graph.*, **28**(3), 38:1–38:8.

Müller M., Charypar D. and Gross M., 2003. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association, 154–159.

Müller M., Heidelberger B., Hennix M. and Ratcliff J., 2007. Position based dynamics. *Journal of Visual Communication and Image Representation*, **18**(2), 109–118.

Müller M., Schirm S. and Teschner M., 2004. Interactive blood simulation for virtual surgery based on smoothed particle hydrodynamics. *Technology and Health Care*, **12**(1), 25–31.

Müller M., Schirm S., Teschner M., Heidelberger B. and Gross M., July 2004. Interaction of fluids with deformable solids: Research articles. *Comput. Animat. Virtual Worlds*, **15**(3-4), 159–171.

Müller M., Solenthaler B., Keiser R. and Gross M., 2005. Particle-based fluid-fluid interaction. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '05, New York, NY, USA. ACM, 237–244.

NASA , 2015. Computational fluid dynamics support for space launch vehicles. `http://www.nas.nasa.gov/SC13/demos/demo5.html` Accessed: 2016-02-17.

NEXT LIMIT , 2014. Nodes - hybrido fluids (HyFLIP). `http://support.nextlimit.com/pages/viewpage.action?pageId=22217265` Accessed: 2016-04-17.

NVIDIA , 2016. CUDA parallel computing. `http://www.nvidia.co.uk/object/cuda-parallel-computing-uk.html` Accessed: 2016-02-17.

NVIDIA, CUDA , 2010. GPU occupancy calculator. *CUDA SDK*.

Ostromoukhov V., July 2007. Sampling with polyominoes. *ACM Trans. Graph.*, **26**(3).

Ostromoukhov V., Donohue C. and Jodoin P.-M., August 2004. Fast hierarchical importance sampling with blue noise properties. *ACM Trans. Graph.*, **23**(3), 488–495.

Öztireli A. C., Alexa M. and Gross M., December 2010. Spectral sampling of manifolds. *ACM Trans. Graph.*, **29**(6), 168:1–168:8.

Öztireli A. C. and Gross M., November 2012. Analysis and synthesis of point distributions based on pair correlation. *ACM Trans. Graph.*, **31** (6), 170:1–170:10.

Pang W.-M., Qu Y., Wong T.-T., Cohen-Or D. and Heng P.-A., 2008. Structure-aware halftoning. In *ACM Transactions on Graphics (TOG)*, volume 27. ACM, 89.

Payne B. A. and Toga A. W., 1992. Distance field manipulation of surface models. *IEEE Computer Graphics and Applications*, **12**(1), 65–71.

Peachey D. R., August 1986. Modeling waves and surf. *SIGGRAPH Comput. Graph.*, **20**(4), 65–74.

Pianthong K., Seehanam W., Behnia M., Sriveerakul T. and Aphornratana S., 2007. Investigation and improvement of ejector refrigeration system using computational fluid dynamics technique. *Energy conversion and Management*, **48**(9), 2556–2564.

Pierre Alliez S. T. and Wormser C., 2015. CGAL: 3D fast intersection and distance computation (AABB tree). `http://doc.cgal.org/latest/AABB_tree/` Accessed: 2016-02-17.

Posner J., Buchanan C. and Dunn-Rankin D., 2003. Measurement and prediction of indoor air flow in a model room. *Energy and buildings*, **35**(5), 515–526.

Premože S., Tasdizen T., Bigler J., Lefohn A. and Whitaker R. T., 2003. Particle-based simulation of fluids. In *Computer Graphics Forum*, volume 22. Wiley Online Library, 401–410.

Raveendran K., Wojtan C. and Turk G., 2011. Hybrid smoothed particle hydrodynamics. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '11, New York, NY, USA. ACM, 33–42.

Reeves W. T., 1983. Particle systemsa technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics (TOG)*, **2**(2), 91–108.

Rigal G., 2015. Fifteen quotes for your 2015 marketing plan and strategy. `http://www.guillaumerigal.com/2015/01/fifteen-quotes-for-your-2015-marketing-plan-and-strategy/` Accessed: 2016-02-17.

Robinson-Mosher A., Shinar T., Gretarsson J., Su J. and Fedkiw R., August 2008. Two-way coupling of fluids to rigid and deformable solids and shells. *ACM Trans. Graph.*, **27**(3), 46:1–46:9.

Rosenhead L., 1931. The formation of vortices from a surface of discontinuity. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, **134**(823), 170–192.

Ryoo S., Rodrigues C. I., Baghsorkhi S. S., Stone S. S., Kirk D. B. and Hwu W.-m. W., 2008. Optimization principles and application performance evaluation of a multithreaded GPU using CUDA. In *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming*. ACM, 73–82.

Satish N., Harris M. and Garland M., May 2009. Designing efficient sorting algorithms for manycore GPUs. In *Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, 1–10.

Schechter H. and Bridson R., July 2012. Ghost SPH for animating water. *ACM Trans. Graph.*, **31**(4), 61:1–61:8.

Schroeder M., 2012. *Fractals, Chaos, Power Laws: Minutes from an Infinite Paradise*. Dover Publications, Incorporated.

Secord A., 2002. Weighted voronoi stippling. In *Proceedings of the 2Nd*

International Symposium on Non-photorealistic Animation and Rendering, NPAR '02, New York, NY, USA. ACM, 37–43.

Seltzerl A., 2013. Tablet effervescing in water. `http://cdn.c.photoshelter.com/img-get/I0000SKjYNOZOxzA/s/860/860/Alka-Seltzer-Tablet-Effervescing-in-Water.jpg` Accessed: 2013-03-10.

Sethian J. A., 1995. A fast marching level set method for monotonically advancing fronts. In *PROC. NAT. ACAD. SCI*, 1591–1595.

Shade J., Cohen M. F. and Mitchell D. P., 2000. Tiling layered depth images. 231–242.

Shcherbacheva A. *Applying fluid mechanics and Kalman filtering to forecasting electricity spot prices*. PhD thesis, Masters Thesis, Lappeenranta University of Technology, Lappeenranta, Finland, 2011.

Shin S.-H., Kam H. R. and Kim C.-H., 2010a. Hybrid simulation of miscible mixing with viscous fingering. *Computer Graphics Forum*, 675–683.

Shin S.-H., Kam H. R. and Kim C.-H., 2010b. Hybrid simulation of miscible mixing with viscous fingering. In *Computer Graphics Forum*, volume 29. Wiley Online Library, 675–683.

Shirley P., 1991. Discrepancy as a quality measure for sample distributions. In *In Eurographics '91*. Elsevier Science Publishers, 183–194.

SideFX , 2016. Houdini. `http://www.sidefx.com/index.php?option=com_content&task=blogcategory&id=117&Itemid=374` Accessed: 2016-02-17.

Sims K., September 1990. Particle animation and rendering using data parallel computation. *SIGGRAPH Comput. Graph.*, **24**(4), 405–413.

Solenthaler B. and Pajarola R., 2008. Density contrast SPH interfaces. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '08, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association, 211–218.

Solenthaler B. and Pajarola R., July 2009. Predictive-corrective incompressible SPH. *ACM Trans. Graph.*, **28**(3), 40:1–40:6.

Solenthaler B. and Gross M., 2011. Two-scale particle simulation. In *ACM SIGGRAPH 2011 Papers*, SIGGRAPH '11, New York, NY, USA. ACM, 81:1–81:8.

Solenthaler B., Schläfli J. and Pajarola R., February 2007. A unified particle model for fluid–solid interactions: Research articles. *Comput. Animat. Virtual Worlds*, **18**(1), 69–82.

SPHERIC , 2016. Spheric grand challenges. `http://spheric-sph.org/grand-challenges` Accessed: 2016-06-17.

Stam J., 1999. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co., 121–128.

Stam J. and Fiume E., 1995. Depicting fire and other gaseous phenomena using diffusion processes. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, New York, NY, USA. ACM, 129–136.

Stomakhin A., Schroeder C., Jiang C., Chai L., Teran J. and Selle A., July 2014. Augmented MPM for phase-change and varied materials. *ACM Trans. Graph.*, **33**(4), 138:1–138:11.

Stora D., Agliati P.-O., Cani M.-P., Neyret F. and Gascuel J.-D., 1999. Animating lava flows. In *Proceedings of the 1999 Conference on Graphics Interface '99*, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc., 203–210.

Subr K. and Kautz J., July 2013. Fourier analysis of stochastic sampling strategies for assessing bias and variance in integration. *ACM Trans. Graph.*, **32**(4), 128:1–128:12.

Sun X., Zhou K., Guo J., Xie G., Pan J., Wang W. and Guo B., July 2013. Line segment sampling with blue-noise properties. *ACM Trans. Graph.*, **32**(4), 127:1–127:14.

Surazhsky V., Surazhsky T., Kirsanov D., Gortler S. J. and Hoppe H., July 2005. Fast exact and approximate geodesics on meshes. *ACM Trans. Graph.*, **24**(3), 553–560.

Tartakovsky A. M., Meakin P., Scheibe T. D. and West R. M. E., 2007. Simulations of reactive transport and precipitation with smoothed particle hydrodynamics. *Journal of Computational Physics*, **222**(2), 654–672.

Taylor A. E., 1952. L'hospital's rule. *The American Mathematical Monthly*, **59**(1), 20–24.

Temam R., 2001. *Navier-Stokes equations: theory and numerical analysis*, volume 343. American Mathematical Soc.

Teschner M., Heidelberger B., Müller M., Pomerantes D. and Gross M. H., 2003. Optimized spatial hashing for collision detection of deformable objects. In *VMV*, volume 3, 47–54.

Trautz M., 1916. Das gesetz der reaktionsgeschwindigkeit und der gleichgewichte in gasen. bestätigung der additivität von cv-3/2r. neue bestimmung der integrationskonstanten und der molekül̈durchmesser. *Zeitschrift für anorganische und allgemeine Chemie*, **96**(1), 1–28.

Ulichney R., 1987. *Digital Halftoning*. MIT Press.

Ulichney R., Jan 1988. Dithering with blue noise. *Proceedings of the IEEE*, **76**(1), 56–79.

University of South Florida , 2016. Holistic numerical methods : Runge-kutta 4th order method. `http://nm.mathforcollege.com/topics/runge_kutta_4th_method.html`.

Volkov V., 2010. Better performance at lower occupancy. In *Proceedings of the GPU Technology Conference, GTC*, volume 10. San Jose, CA, 16.

Wachtel F., Pilleboue A., Coeurjolly D., Breeden K., Singh G., Cathelin G., de Goes F., Desbrun M. and Ostromoukhov V., July 2014. Fast tile-based adaptive sampling with user-specified Fourier spectra. *ACM Trans. Graph.*, **33**(4), 56:1–56:11.

Walsh M., Wallis F. and Grace P., 2005. 3-D numerical simulation of blood flow through models of the human aorta. *Journal of biomechanical engineering*, **127**, 767.

Wei L.-Y., August 2008. Parallel Poisson disk sampling. *ACM Trans. Graph.*, **27**(3), 20:1–20:9.

Wei L.-Y., July 2010. Multi-class blue noise sampling. *ACM Trans. Graph.*, **29**(4), 79:1–79:8.

Wei L.-Y. and Wang R., July 2011. Differential domain analysis for non-uniform sampling. *ACM Trans. Graph.*, **30**(4), 50:1–50:10.

Weißmann S., Pinkall U. and Schröder P., July 2014. Smoke rings from smoke. *ACM Trans. Graph.*, **33**(4), 140:1–140:8.

Whittaker E. T., 1935. On Gauss' theorem and the concept of mass in general relativity. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, **149**(867), 384–395.

Wojtan C., Carlson M., Mucha P. J. and Turk G., 2007. Animating corrosion and erosion. In *Proceedings of the Third Eurographics conference on Natural Phenomena*, NPH'07, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association, 15–22.

Xu R., Stansby P. and Laurence D., 2009. Accuracy and stability in incompressible SPH (ISPH) based on the projection method and a new approach. *Journal of Computational Physics*, **228**(18), 6703–6725.

Xu Y., Hu R., Gotsman C. and Liu L., 2012. Blue noise sampling of surfaces. *Computers and Graphics*, **36**(4), 232 – 240. Applications of Geometry Processing.

Xu Y., Liu L., Gotsman C. and Gortler S. J., 2011. Capacity-constrained delaunay triangulation for point distributions. *Computers and Graphics*, **35**(3), 510 – 516. Shape Modeling International (SMI) Conference 2011.

Yang Q., Jones V. and McCue L., 2012. Free-surface flow interactions with deformable structures using an sph–fem model. *Ocean Engineering*, **55**, 136–147.

Yang T., Chang J., Ren B., Lin M. C., Zhang J. J. and Hu S.-M., October 2015. Fast multiple-fluid simulation using Helmholtz free energy. *ACM Trans. Graph.*, **34**(6), 201:1–201:11.

Yue Y., Smith B., Batty C., Zheng C. and Grinspun E., November 2015. Continuum foam: A material point method for shear-dependent flows. *ACM Trans. Graph.*, **34**(5), 160:1–160:20.

Zhang Y., Solenthaler B. and Pajarola R., 2007. GPU accelerated SPH particle simulation and rendering. In *ACM SIGGRAPH 2007 Posters*, SIGGRAPH '07, New York, NY, USA. ACM.

Zheng W., Zhu B., Kim B. and Fedkiw R., 2015. A new incompressibility discretization for a hybrid particle MAC grid representation with surface tension. *Journal of Computational Physics*, **280**, 96–142.

Zhou Y., Huang H., Wei L.-Y. and Wang R., July 2012. Point sampling with general noise spectrum. *ACM Trans. Graph.*, **31**(4), 76:1–76:11.

Zhu G. and Kronast M., 1993. The calculation of ground effect on a car flow field using two dimensional Navier-Stokes equations. *Acta Aerodynamica Sinica*, **11**(1), 22–32.

Zhu Y. and Bridson R., July 2005. Animating sand as a fluid. *ACM Trans. Graph.*, **24**(3), 965–972.

Zocco D., 2007. Turbulent hydrodynamics and the stock market: the art of creating a fokker-planck equation from a system with high kurtosis. `http://www.physics.ucsd.edu/~dzocco/research/papers/fokpla_dzocco2007.pdf` Accessed: 2016-02-17.